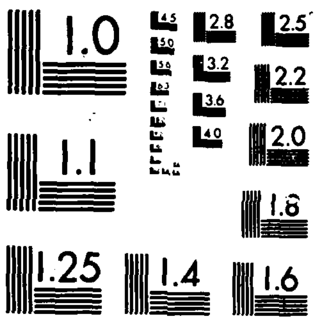
AD-A164 152 MICROPROCESSOR CONTROLLER WITH NONVOLATILE MEMORY 1/2
IMPLEMENTATION(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA
J W WALLIN DEC 85

UNCLASSIFIED F/G 9/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

(2)

AD-A164 152

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MICROPROCESSOR CONTROLLER WITH
NONVOLATILE MEMORY IMPLEMENTATION

by

Jay Weston Wallin

December 1985

Thesis Advisor: R. Panholzer

AD-A164 152

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5100	7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5100
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO. / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
MICROPROCESSOR CONTROLLER WITH NONVOLATILE MEMORY IMPLEMENTATION

12 PERSONAL AUTHOR(S)
Wallin, Jay W.

13a TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ________ TO ________	1985 December	130

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Microprocessor Controller; Bubble Memory; NSC800

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

In support of the Naval Postgraduate School's space program, a small, self-sufficient, low power microprocessor controller with nonvolatile memory has been designed, constructed and tested. Because of limited battery power availability, Complementary Metal-Oxide Semiconductor (CMOS) components have been used. The controller uses the National Semiconductor NSC800 CPU along with three NSC810A RAM-I/O-Timers. Other features include a 16 channel analog-to-digital converter, a real time clock, local memory in the form of EPROM and RAM, and the Intel BPK 72 Bubble Memory System. The general nature of the controller allows it to

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL R. Panholzer	22b TELEPHONE (Include Area Code) 408 646-2154	22c. OFFICE SYMBOL 62Pz

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

be reprogrammed and utilized in a variety of applications. Prior thesis research by Captain Mike Snyder, U.S. Army, using the NSC888 Self-Contained NSC800 Evaluation System [Ref. 1] has been expanded upon in this thesis project.

Microprocessor Controller with Nonvolatile Memory Implementation

by

Jay Weston Wallin
Lieutenant, United States Navy
B.S., United States Naval Academy, 1979

Submitted in partial fullfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December, 1985

Author: [illegible]
Jay Weston Wallin

Approved by: [illegible]
R. Panholzer, Thesis Advisor

[illegible]
C.H. Lee, Second Reader

[illegible]
H.B. Rigas, Chairman, Department of Electrical and Computer Engineering

[illegible]
J.N. Dyer, Dean of Science and Engineering

ABSTRACT

In support of the Naval Postgraduate School's space program, a small, self-sufficient, low power microprocessor controller with nonvolatile memory has been designed, constructed and tested. Because of limited battery power availability, Complementary Metal-Oxide Semiconductor ~~(CMOS)~~ components have been used. The controller uses the National Semiconductor NSC800 CPU along with three NSC810A RAM-I/O-Timers. Other features include a 16 channel analog-to-digital converter, a real time clock, local memory in the form of EPROM and RAM, and the Intel BPK 72 Bubble Memory System. The general nature of the controller allows it to be reprogrammed and utilized in a variety of applications. Prior thesis research by Captain Mike Snyder, U.S. Army, using the NSC888 Self-Contained NSC800 Evaluation System [Ref. 1] has been expanded upon in this thesis project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I wish here to express my thanks to John Glenn and David Rigmaiden for their technical support. I also would like to thank Mike Reid for his help in thesis documentation.

I. BACKGROUND AND INTRODUCTION

A. GETAWAY SPECIAL (GAS) PROGRAM

The Space Shuttle transportation system further opens the frontier of space for educational, scientific, and industrial purposes. This is the result of a relatively low cost space vehicle and its variable payload. In 1976 NASA responded to the needs of the scientific, educational and industrial community, and established the Get Away Special (GAS) program. Under this program, free space in the shuttle's cargo bay, following the scheduling of primary payloads, is assigned to self-contained GAS experiments, thus increasing the number of government and civilian experiments possible. Hopefully, the knowledge gained will be applicable to future GAS projects.

B. WHAT PROMPTED SPACE PROJECT

On early missions, Space Shuttle cargo bay experiments were plagued by minor crystalline and circuit board breakage. The causes of this damage were not positively identified and were attributed to factors ranging from low frequency structural resonance in the cargo bay to outgassing through cargo bay vents during launch. In an attempt to further isolate the causes of this breakage, the GAS experiment, which this controller supports, will collect acoustic data near a suspect vent during launch. In order to establish a baseline for later analysis, a sound signature of the shuttle bay will be taken prior to launch. The controller will insure that these measurements and other events are performed in the correct sequence.

C. GENERAL REQUIREMENTS OF GAS EXPERIMENT

The GAS general requirements, as described in the _Get Away Special (GAS) Small Self-Contained Payloads Experimenter Handbook_ [Ref. 2] will be expanded upon in the following discussion. The requirements concern the three functional areas: self containment, safety, and shuttle environment.

GAS experiments must deliver their own power within the enclosure provided. This self containment necessitates some type of internal power distribution and a minimum of power consumption. All system control during the lifetime of the experiment is also provided inside the GAS container. Experiment initiation is the only exception. Any data retrieval and storage must be done within the container.

All components utilized in the experiment must be of safe construction and not pose any risk of damaging the shuttle or any other experiment. The controller can actually improve experiment safety by identifying possible electrical faults and isolating them.

An understanding of the shuttle environment is necessary to ensure proper system component utilization and construction. The environmental effects during launch and return can be grouped into four general areas: acoustics, random vibration, acceleration and temperature. As reported in the Experimenter's Handbook, the primary environmental effects to be considered are random vibration and acceleration. These launch and return conditions are presented in Figure 1. Thermal considerations are of lesser importance, due to enclosure insulation and the short duration of the experiment, but are worthy of mention. Temperature can have an effect if component tolerances are excessively temperature dependent. One heat source to consider is the shuttle bay environment while the experiment is waiting for launch and during the flight. Another heat source is the temperature due to controller system power dissipation. The system's primary CMOS construction renders this less important.

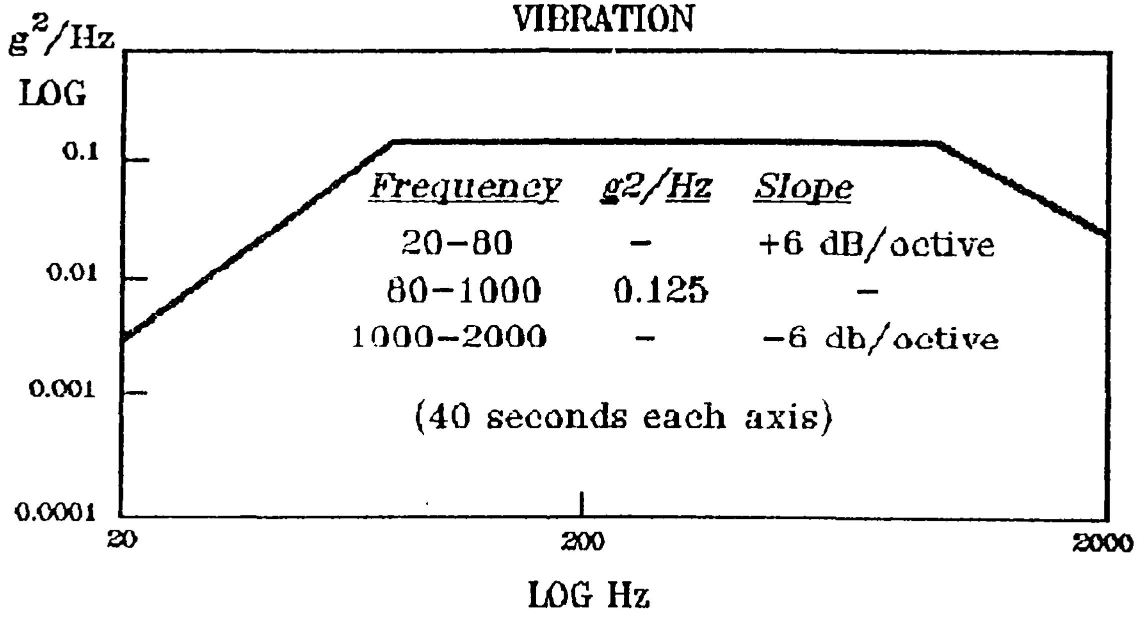
D. UNIQUE REQUIREMENTS ASSOCIATED WITH EXPERIMENT

First and foremost is the requirement of self sufficiency. The controller's central processor must have the capacity to control all aspects of system operation. As was mentioned in the previous chapter, the power requirements of the experiment must be contained within the enclosure provided. With a limited amount of room to hold overall system power, a number of unique approaches must be incorporated.

SHUTTLE ENVIRONMENT

RANDOM

VIBRATION



Frequency	*g2/Hz*	*Slope*
20–80	–	+6 dB/octive
80–1000	0.125	–
1000–2000	–	–6 db/octive

(40 seconds each axis)

Accelerations

Quasi–Steady State

Limit Load Factors

Across Container Axis = + 6.0 g's	Overall Root Mean Squared
Along Container Axis = + 10.0 g's	Random Vibration Level is 12.9 g's

Figure 1. Launch and Return Conditions

The most obvious power saving technique is the use of CMOS devices wherever possible. Therefore, the microprocessor used for the controller and its supporting components should be of CMOS type. Next, by reducing the clock frequency, the power dissipated in the system will be reduced proportionally. The use of power-down features can also significantly reduce system power consumption. With the inclusion of a power-down operation, a real time reference is required to ensure that the system is powered up at the right moment.

The need for multiple unit control is apparent in the GAS experiment. With all system controls internal, the jobs of experiment initiation, system status and record keeping fall on the microprocessor and its supporting components. A real time clock is necessary to turn devices on and off at specific instances, and to control the duration of the experiment. Multiple unit control also means multiple paths to and from various components. These multiple paths equate to an extended number of input and output ports on the controller microprocessor system. An analog-to-digital interface must be available to monitor system functions such as temperature and bus voltage. The number of sensors monitored determine the number of channels utilized.

Finally the question of reliability must be looked at. With the experiment controlled completely within the GAS canister, some degree of feedback is required to insure that the system is operating properly. For instance, when an experiment is powered up, indications to that effect should be returned to the microprocessor. Error detection and correction should also be addressed.

II. HARDWARE SELECTION

A. MICROPROCESSOR SYSTEM

1. NSC800

Controller operation is based on the National Semiconductor's NSC800 family. The NSC800's block diagram is provided in Figure 2. The following discussion is a summary of the features that made this selection attractive. More detailed information on the NSC800's operation is found in the associated data sheet [Ref. 3]. The first advantage in using this microprocessor is its CMOS construction. As previously mentioned, the environment in which the controller will be operating necessitates low power usage. Another feature that makes the NSC800 a viable microprocessor for the controller is the Z-80 instruction set it supports. The lab environment available during the initial phases of the space project also support Z-80 and 8080 development. The natural question to ask is, "Why not simply use a Zilog Z-80 chip?" The CMOS version of the Zilog Z-80 chip was not available during the initial design phase of the project. Like the Intel 8085, the National NSC800 microprocessor also has the ability to multiplex the address/data bus. Although the NSC800 is an 8-bit processor, an effective 16-bit address can be achieved. The 16-bit address is formed by multiplexing the lower address lines (A0-A7), latching them

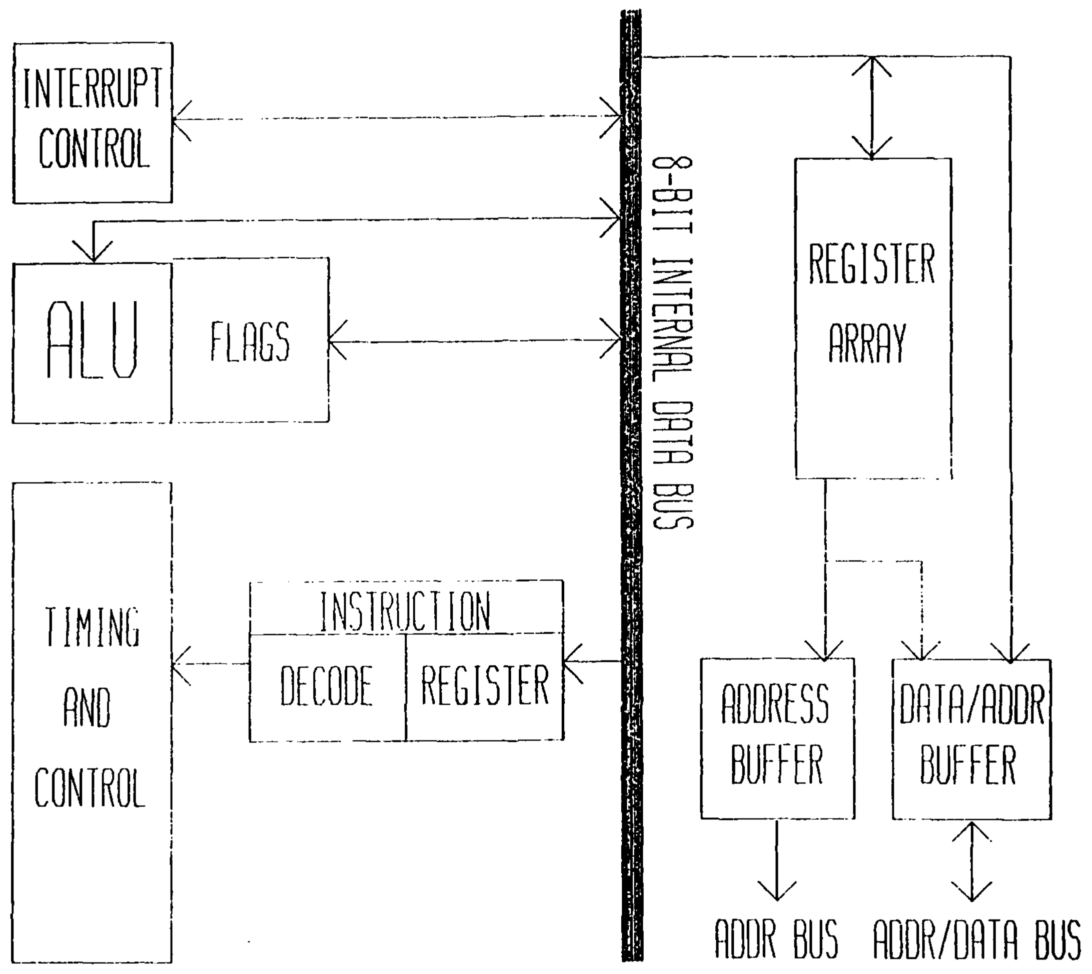


Figure 2. NSC800 Block Diagram

externally, and combining them with the upper non-multiplexed address buss (A8-A15). This equates to an address space of 64K. The power supplied to the NSC800 does not have to be excessively regulated. The microprocessor can operate with a voltage ranging from 2.4 to 6.0 volts. This experiment will use a nominal voltage supply of 5.0 volts. However, there are tradeoffs that occur if voltage is maintained at the minimum of 2.4 volts. While a savings of power will occur, the maximum clock frequency of the NSC800 will be limited to 500 KHz. The operational frequency selected for the controller is 4 MHz, which yields an internal instruction cycle of 1.0 microseconds. The extended I/O requirements imposed and the capability of addressing up to 256 I/O devices make this microprocessor appealing.

A pin level view of the NSC800 follows. The NSC800 chip pinout is detailed in Figure 3. Through this exploration of the microprocessor, a better understanding of overall controller operation will be achieved. The first 8 pins of the NSC800 make up the upper byte for memory addressing (A8-A15). During I/O operations this byte replicates the lower address byte (A0-A7). This is the reason why memory can be addressed up to 65536 bytes and I/O can only be addressed up to 256. Next is pin 9, the clock output. This provides a system time reference and operates at one half the input

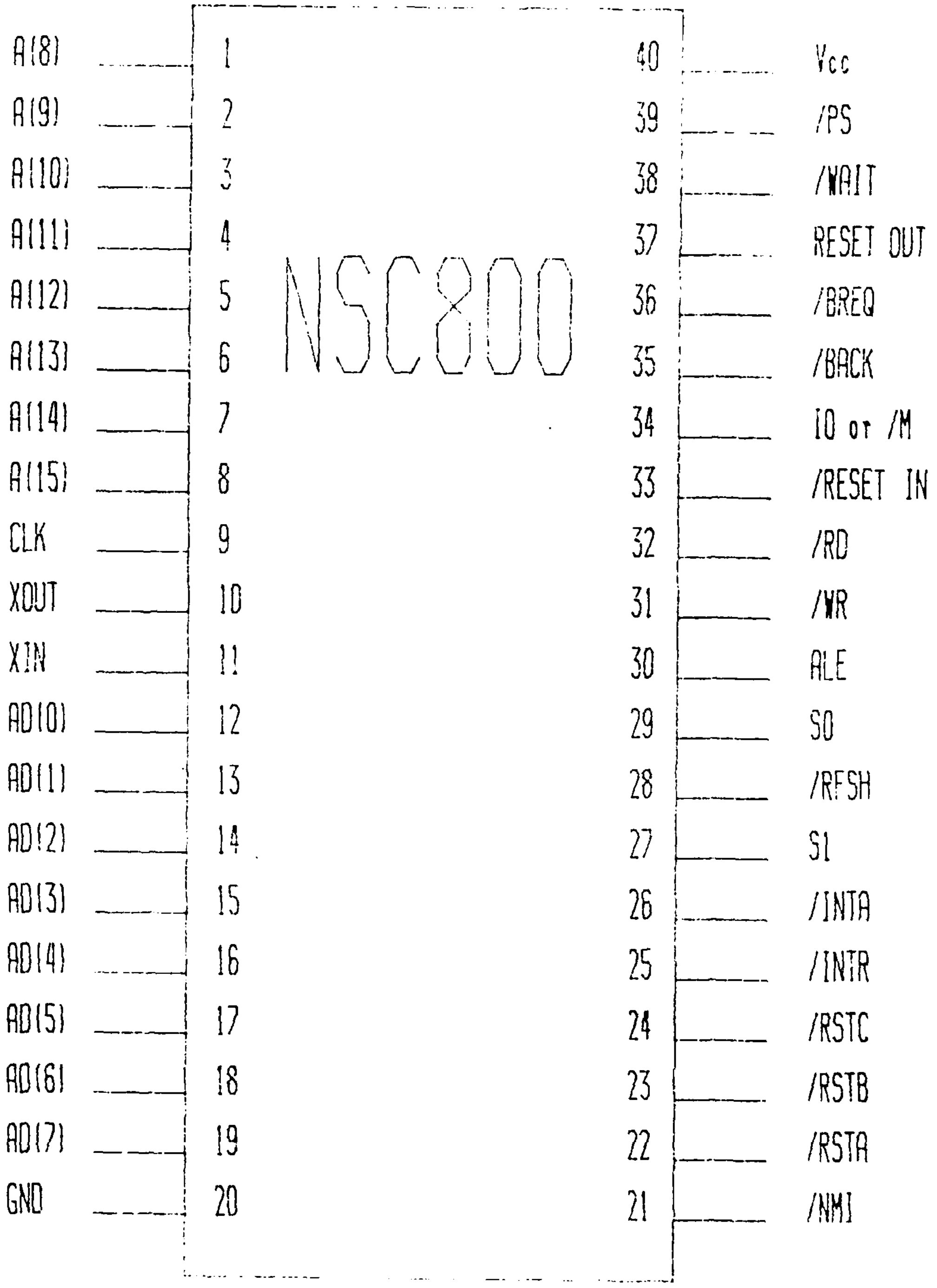


Figure 3. NSC800 Chip Pinout

clock frequency. In this application the clock output will be 2 MHz. The XOUT and XIN pins are used for frequency input to the NSC800. The circuit interface between the 4MHz crystal and the NSC800 is provided in Figure 4. The next 8 pins make up the lower address/data bus. As previously discussed, this bus is multiplexed. The lower byte is formed by first putting the lower address on the bus. The address lines are then latched to the bus through external circuitry and the Address Latch Enable (ALE) line. This circuit is described in Figure 5. Pins 21 through 26 are the NSC800's interrupt control lines. The interrupts are both mask and non-maskable and allow a varied degree of control. NSC800 status (S0 and S1) is provided on pins 27 and 29. Table 1 demonstrates the relationship between the status bits and system operation. The microprocessor also has the capabil

TABLE 1. STATUS BITS AND SYSTEM OPERATION

OPERATION	S0	S1	IO/M	RD	WR
OPCODE FETCH	1	1	0	0	1
MEMORY WRITE	1	0	0	1	0
MEMORY READ	0	1	0	0	1
I/O WRITE	1	0	1	1	0
I/O READ	0	1	1	0	1
INTERNAL OPS	0	1	0	1	1
INTERRUPT ACK	1	1	0	1	1
HALT	0	0	0	0	1

ity of being reset through pin 33. An output reset for supporting external peripherals is provided on pin 37. The question ,"Is data going to memory or I/O ?" can be answered with pin 34, the IO/M select line. Additional lines include

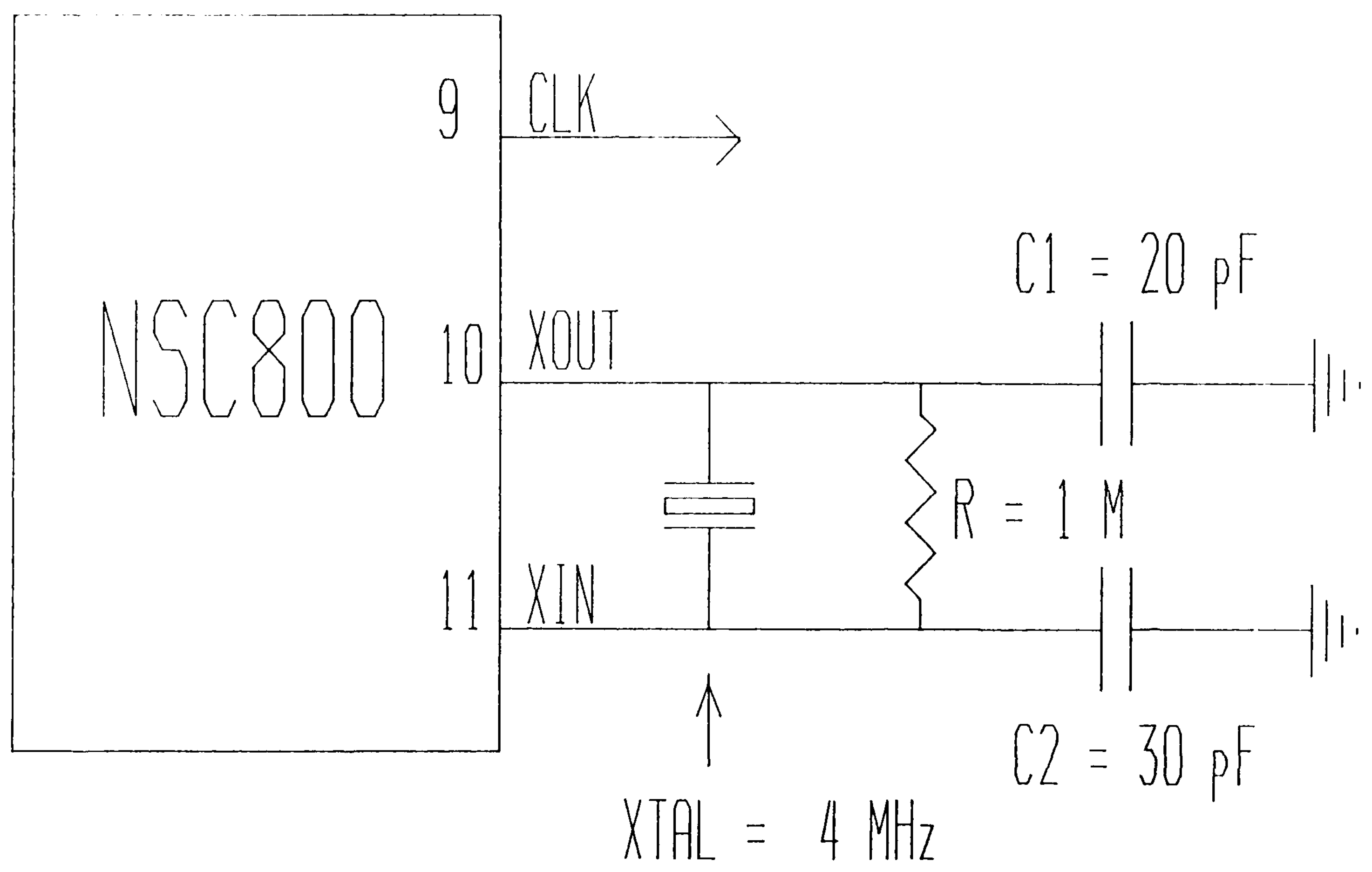


Figure 4. NSC800 Oscillator Circuitry

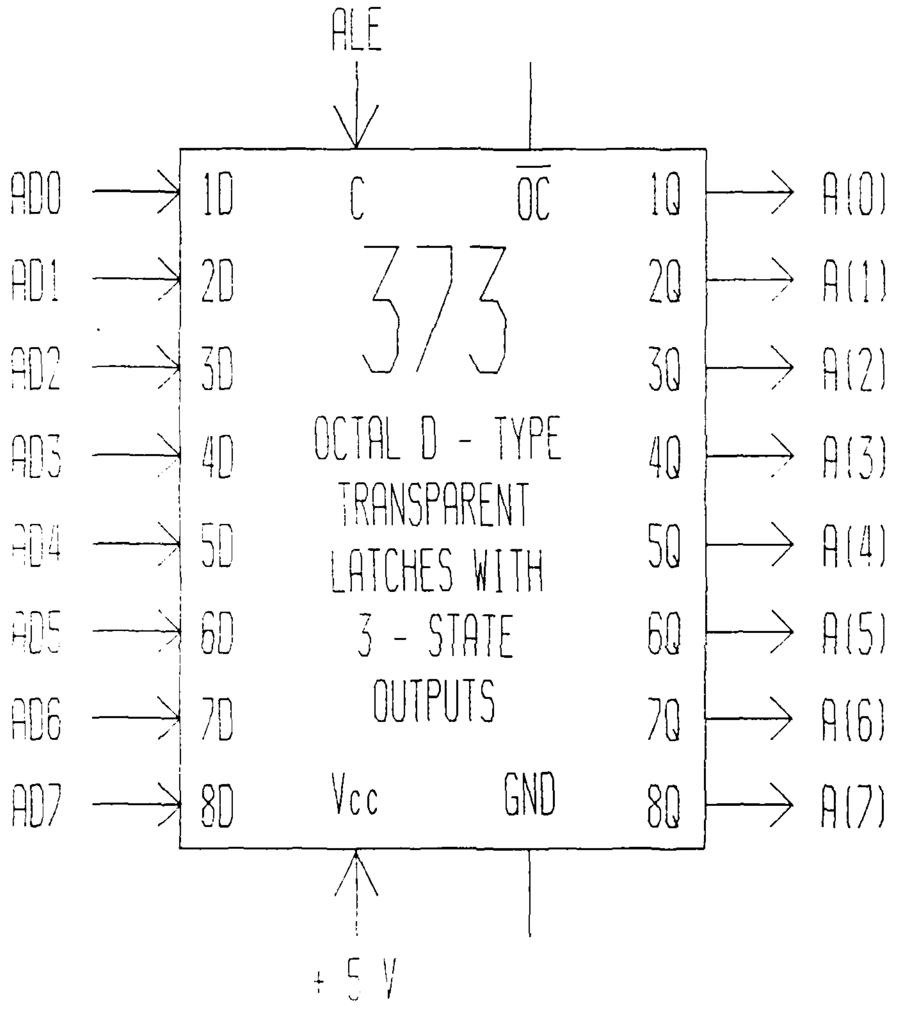


Figure 5. Lower Address Latch Circuitry

read, write and chip ground. For I/O operations, which might last longer than the I/O hold time, there is a microprocessor wait (/WAIT) on pin 38. This will allow additional t states to occur, until it is deactivated. Some features not utilized in this configuration are DMA, memory refresh, and power-down.

There are a number of tradeoffs which have to be addressed when selecting which features of the NSC800 will be used and which ones will not be used. Some of these have already been discussed. Other tradeoffs occur in the use of dynamic or static RAM. By using static RAM, less power will be required for memory storage operations. One problem associated with static RAM is its larger package size compared to similar dynamic RAM. Tradeoffs also must be reckoned with in not utilizing DMA in the controller design. Although overall system design is simplified, flexibility is reduced because certain polled operations tie up the microprocessor.

2. NSC810A

The NSC810 RAM-I/O-Timer is chosen for system I/O control as well as some controller clock and timer requirements. Specific information on the NSC810A RAM-I/O-Timer is found in the associated data sheet [Ref. 4]. The NSC810A system block diagram is provided in Figure 6. This is the natural choice since it belongs to the NSC800 family.

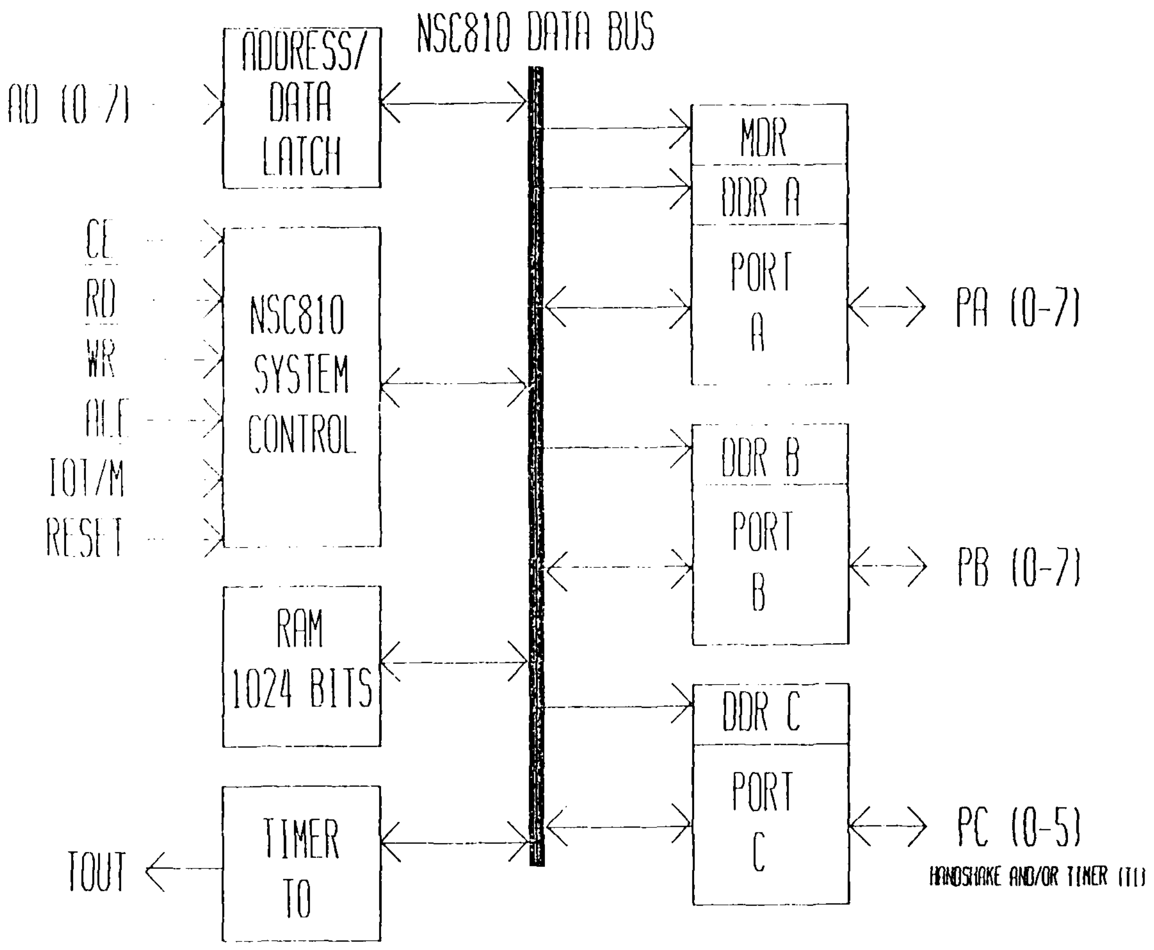


Figure 6. NSC810A Block Diagram

Another favorable feature is its CMOS construction. As is the case with the NSC800 microprocessor, a wide range of input voltage is acceptable to the chip. The range is 2.4 to 6.0 volts. Extended control of I/O is available through three programmable ports. Specific information on these ports will be presented in a following chapter. A very broad range of programmable clock and timer modes is an additional feature that makes this choice very interesting.

A review of some of the pin level aspects of the NSC810 follow. The NSC810A chip pinout is provided in Figure 7. Many of the pin assignments, similar in nature to the NSC800, will not be discussed. The NSC810 ports are found on pins 21 through 39 and pins 1,2 and 5. These ports are broken up into three groups labeled A, B and C. The first is Port A. Port A is the most versatile of the three and can be set up for basic input/output operations or one of three strobed modes. Port B can only be used in a basic input/output mode. The final subsection is port C. This port plays a triple role. It can be set up for basic I/O, or can be used to support handshaking when port A is set up for strobed operation, or can provide a programmable timer output. The C port provides the second NSC810 programmable clock/timer output (T1) available on the NSC810. The primary NSC810 programmable timer/clock output, pin 6, is called "T0". This is the main clock output from the NSC810.

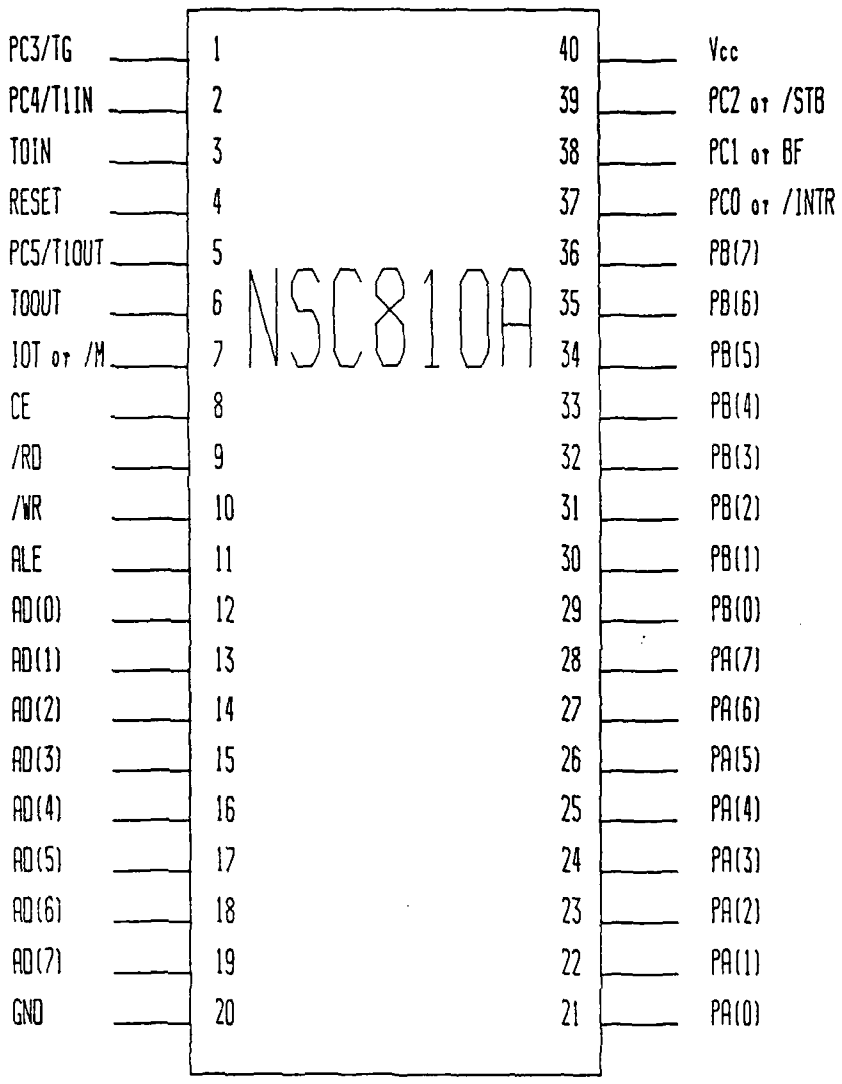


Figure 7. NSC810A Chip Pinout

Both clock outputs can be set up in one of six modes of operation. The NSC810 also has an I/O Timer and Memory arbitrator (IOT or M) at pin 7. This line allows selection of either the chip's timer function or memory on the NSC810. Each NSC810 has 1024 bits of static RAM.

The NSC810 ports are used to control associated external equipment. These ports have three programmable port assignment features: port bit manipulation, port mode assignment, and input or output assignment. The individual port bits can be manipulated with the bit-set and bit-clear functions. As the names imply, the bit-set operation will set a selected bit to a logic level of "1" and the bit-clear function will clear a selected bit to "0". The actual bit in the port that is acted upon is identified by an input mask. An example of the bit-set and clear operation is provided in Figure 8. Next there is the port mode assignment. This takes place in the mode definition register. The four assignments that can be selected are basic I/O, strobed input, strobed output, and strobed output with a tri-state feature. Handshaking is found in the strobed modes. If connected to smart peripherals, the handshaking capability might free up microprocessor operation by not having to continually check a port for status information. The mode definition register configurations are provided in Figure 9. The data direction register determines whether a port is set up as an input or as an output. By selecting the data direction register for

OPERATION	BIT - SET A1, A3, A5	BIT - CLEAR A2, A4
ADDRESS	0 0 0 0 1 1 0 0	0 0 0 0 1 0 0 0
DATA	0 0 1 0 1 0 1 0	0 0 0 1 0 1 0 0
PORT OUTPUT		
BEFORE	1 1 0 1 1 1 0 0	1 1 1 1 1 1 0 0
AFTER	1 1 1 1 1 1 1 0	1 1 1 0 1 0 0 0

Figure 8. Example of Bit-Set and Bit-Clear Operation

MODE 0 - BASIC I/O

MODE 1 - STROBED MODE INPUT

MODE 2 - STROBED MODE OUTPUT (ACTIVE)

MODE 3 - STROBED MODE OUTPUT (TRI-STATE)

Figure 9. Mode Definition Configurations

a particular port and inputting a "1" or "0", it will be set up as an output or input respectively.

Programmable timer and clock operations on the NSC810, are very versatile. The clock is actually a 16-bit up down counter and can be used in any one of six different modes. The six modes of operation range from using it as an event counter to providing a square wave output. The controller will use the square wave output. To control the duration of the square wave clock period, a modulus register is used along with a prescale function if desired. A start and stop clock function is also available.

B. BUBBLE MEMORY

Before going into specific details of the bubble memory device used with the controller, a brief overview of bubble memory fundamentals is in order. The bubble memory system used on the controller functions somewhat like a disk drive and has a capacity of 1 Mbits. That equates to 128K of 8 bit words or bytes. Memory is divided up into blocks or pages. Each page in memory comprises 64 bytes. Over two thousand pages of memory are available on the bubble memory chip. Non-volatility is a unique feature of the bubble memory system. When the system is powered down, either on purpose or by system failure, bubble memory data will not be lost. The power supplied must satisfy certain decay rate conditions in order to ensure the non-volatile nature of the bubble memory. The 5 volt source must have a decay rate of at least 1.1 volts per millisecond and the 12 volt supply must have a decay rate of at least .45 volts per millisecond. Specific information on bubble memory operation is provided in the _BPK 72 Bubble Memory Prototype Kit User's Manual_ [Ref. 5]. The bubble memory system component layout and block diagram are provided in Figures 10 and 11. These diagrams will provide information on system interaction and the bubble-to-controller interface.

After the bubble memory system is selected one of two sets of registers is available. These two registers are chosen with the lower address bus' A0 line as shown in the bubble memory system's port diagram in Table 2. When the A0

TABLE 2. BUBBLE MEMORY PORT ASSIGNMENTS

Address (Hex)	Read (R)/ Write (W)	Assignment
		BMC Register Selection
80		A0 = 0 : BMC FIFO Selection
81		A0 = 1 : Command Register, Status Register, or Register Address Counter Selected

line is at a logic "0" level, the FIFO or parametric registers can be written to. The parametric registers tell the bubble memory system how large a data block to be sent and at what area in bubble memory it will begin. When the A0 line is at a logic level of "1" the command register, status

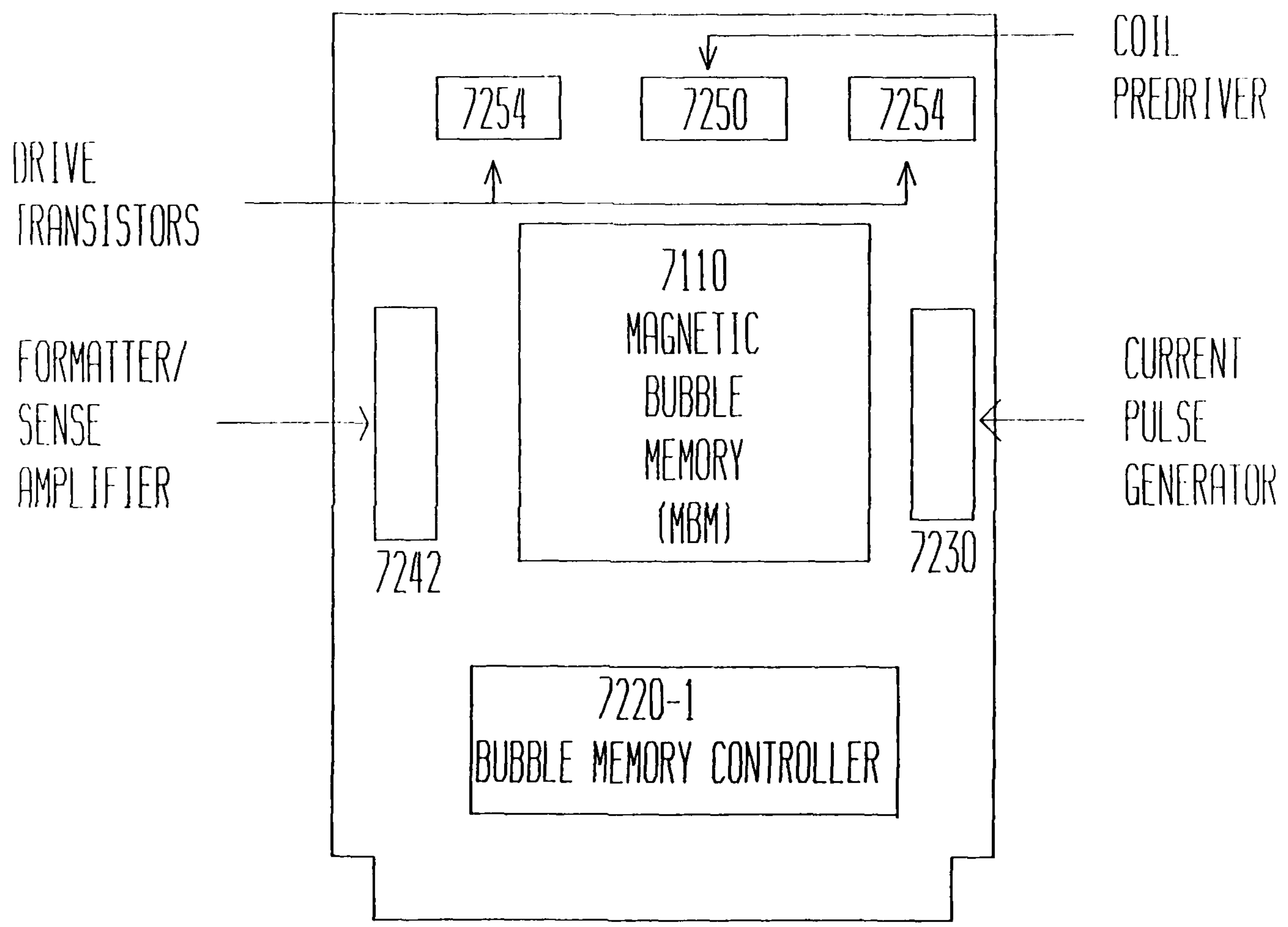

COIL PREDRIVER
7254
7250
7254
DRIVE TRANSISTORS
7110 MAGNETIC BUBBLE MEMORY (MBM)
FORMATTER/ SENSE AMPLIFIER
CURRENT PULSE GENERATOR
7242
7230
7220-1 BUBBLE MEMORY CONTROLLER

Figure 10. Bubble System Component Layout

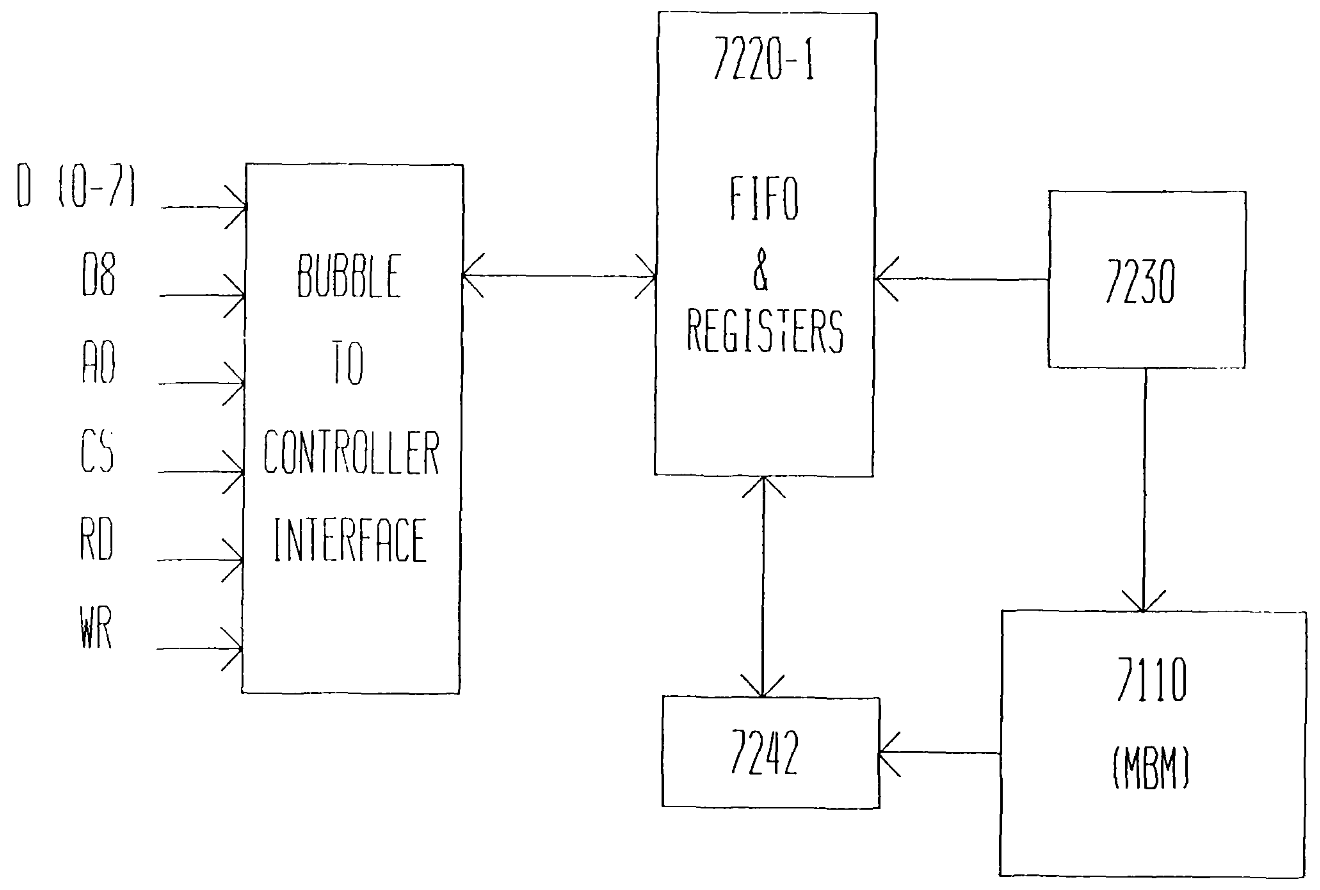


Figure 11. Bubble System Block Diagram

register, or the register address counter are used in bubble memory system operations. Seven of the bidirectional data lines are used to transfer information to and from the bubble memory system. The eighth data line, D8, it selected, is used for parity checking. The bubble memory utilizes only an odd parity check option. There are a number of options not utilized on this particular bubble memory system. The first feature not utilized is Direct Memory Access or DMA. Because of this, the DMA acknowledge pin will be tied to a high logic level. Interrupt is another data transfer mode available but not used for this bubble memory system. Although the microprocessor might be freed up to perform other tasks if the data transfer interrupt feature is used, overall controller system complexity would increase. In this controller system application, the bubble memory operates exclusively as a polled device. Another pin not used in this controller configuration is the 7472 chip select line. With single bubble memory applications this line is tied to a low logic level. If a multiple bubble memory system were utilized, this line would be manipulated to control the byte size going into the bubble memory.

Special power down requirements, besides that of the decay rate of the power supply, have to be addressed. The bubble memory device will be powered down when not in use. Although the bubble memory has an initialization time of up to 160 msec, the duty cycle of writing to the bubble will not be exceeded. Since the bubble memory will be by far the greatest source of power dissipation in the controller system, it will be turned off when not in use.

C. ANALOG-TO-DIGITAL CONVERTER

The controller's analog-to-digital converter has a number of features that make it attractive. The National Semiconductor ADC0816 Single Chip Acquisition System is an 8-bit microprocessor-compatible analog-to-digital converter with 16 channels of analog input. These voltage channel inputs are selected through a local multiplexer that utilizes 4 outside address lines labeled A through D. The actual port assignments associated with the various A-to-D channels is presented in Table 3. A block diagram and pin level discussion of the converter is provided in Figures 12 and 13. This should help in the following description of the converter operation.

The converter's CMOS construction is the first important feature, since low power dissipation is important. Secondly, the converter can handle up to 16 channels of analog data. This feature should more than compensate for the anticipated growth that will occur in the experiment that this controller supports. The converter uses a successive approximation technique for conversion. The approximation

TABLE 3. A-TO-D CONVERTER PORT ASSIGNMENTS

Address (Hex)	Read (R)/ Write (W)	Assignment
E0		Channel # 0
E1		Channel # 1
E2		Channel # 2
E3		Channel # 3
E4		Channel # 4
E5		Channel # 5
E6		Channel # 6
E7		Channel # 7
E8		Channel # 8
E9		Channel # 9
EA		Channel # 10
EB		Channel # 11
EC		Channel # 12
ED		Channel # 13
EE		Channel # 14
EF		Channel # 15

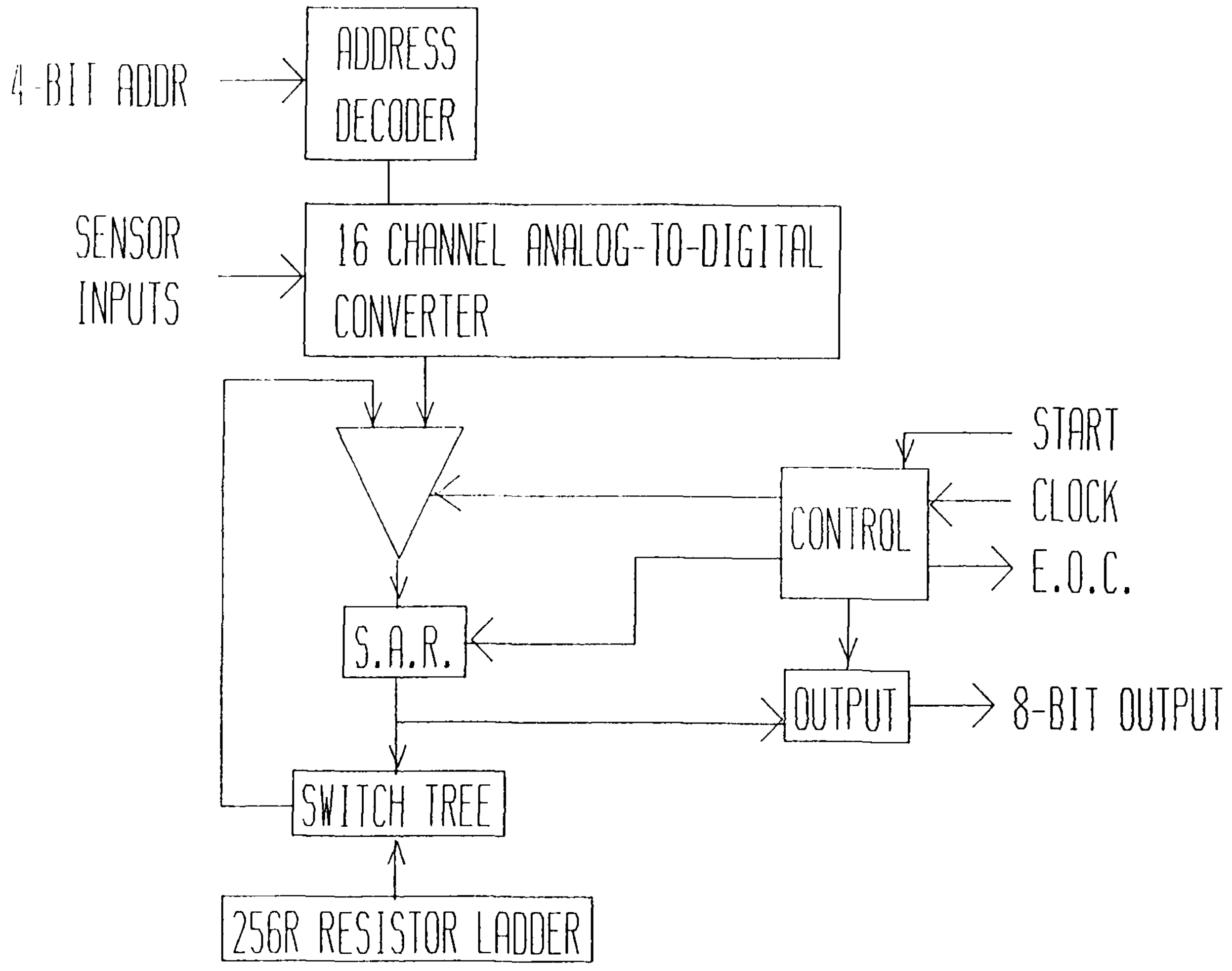


Figure 12. Analog-to-Digital Converter Block Diagram

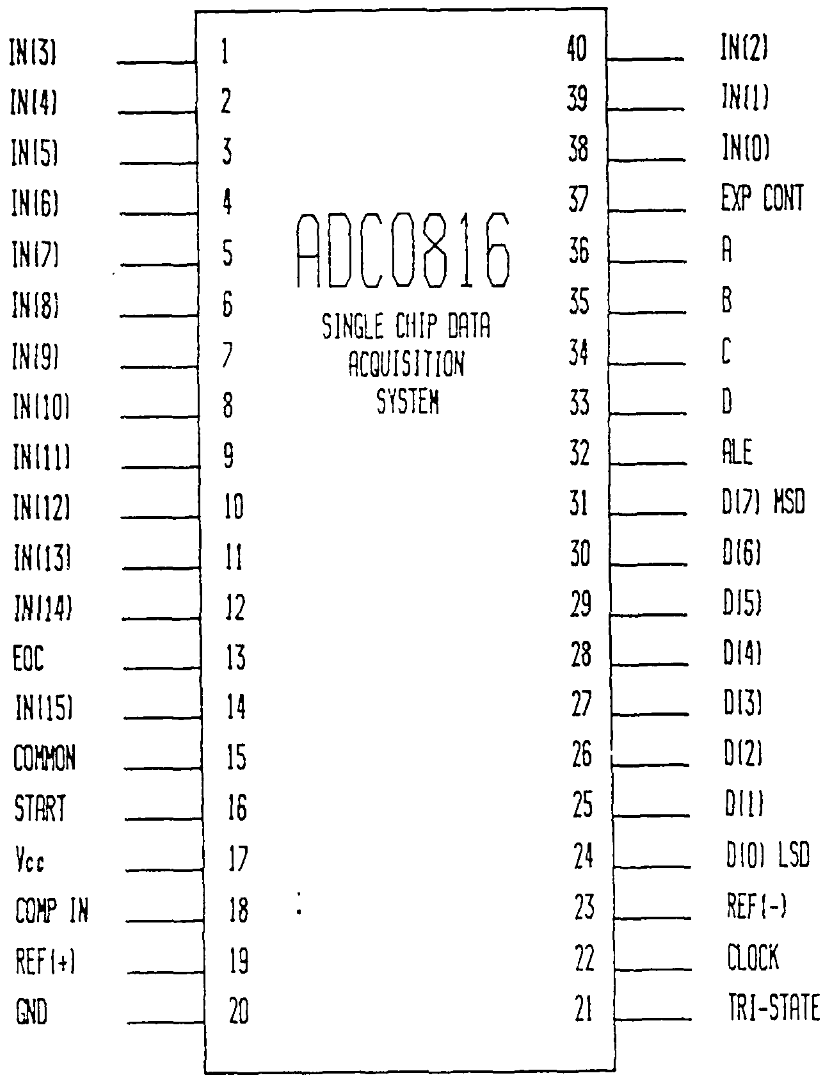


Figure 13. Analog-to-Digital Converter Chip Pinout

technique is divided into three areas: a 256R ladder network, a successive approximation register and a comparator. The 256R ladder network and successive approximation register work together to determine the channel voltage input. The input voltage is compared to the voltage in the ladder eight times to determine the voltage on the channel. The actual tree structure utilized is shown in Figure 14. A chopper-stabilized comparator is used in the converter. The input DC voltage is converted to an AC signal, this signal is then passed through a high gain AC amplifier and the DC level is restored. Why convert a DC input to AC, and then back as DC? This technique removes a DC component in the input signal which causes drift. By removing this drift component the converter is less affected by temperature fluctuations and long term drift. The analog-to-digital converter will be used for ratiometric purposes in this application. The voltage being measured can be represented as a percentage of a full scale value. The voltage on the A-to-D channel can be described by the equation in Figure 15. Some of the most favorable A/D conversion techniques are packaged together in this conversion process. These combine to make this converter highly accurate, repeatable, very tolerant of temperature variation, and extremely fast. The typical conversion time of this converter is 100

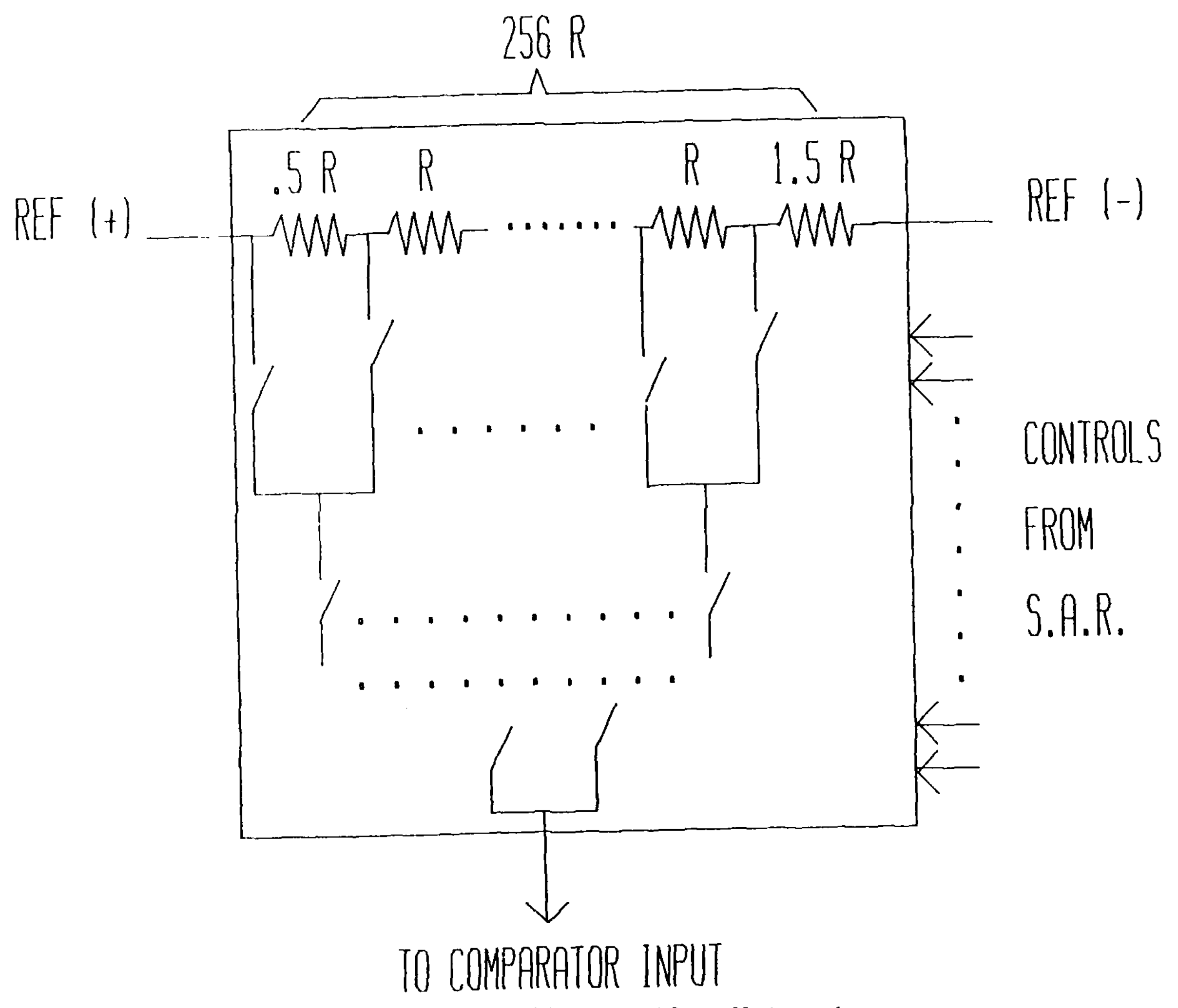


Figure 14. 256R Ladder Network

$$\frac{V_{IN}}{V_{fs} - V_{z}} = \frac{D_{X}}{D_{MAX} - D_{MIN}}$$

V_{IN}	=	INPUT VOLTAGE INTO THE A-TO-D CONVERTER
V_{fs}	=	FULL-SCALE VOLTAGE
V_{z}	=	ZERO VOLTAGE
D_{X}	=	VALUE BEING MEASURED
D_{MAX}	=	MAX VALUE LIMIT
D_{MIN}	=	MIN VALUE LIMIT

Figure 15. Ratiometric Equation

microseconds. Specific information on chip operation is provided in the National Semiconductor's ADC0816 component data sheet [Ref. 6].

D. UART

With a controller-to-"dumb" terminal interface comes a need to control its inherent asynchronous serial data path. To manage this, some type of serial to parallel device must be used. A number of options were investigated. The first of which is using the NSC800's microprocessor serial input. External serial data would be input directly into the controller system through the NSC800. One problem associated with this procedure is the excessive work load the NSC800 would incur to control this input. The next alternative investigated uses a specialized Universal Asynchronous Receiver Transreceiver (UART) and an external baudrate generator to handle the receive and transmit functions. The UART's receiver converts serial channel data to a parallel data format compatible with the controller's internal data bus. The UART can also transmit parallel data that comes off of the controller's system data bus to an external serial data line. The selection of receive and transmit operation is handled by the controller's NSC800 microprocessor. The UART ports that are affected by this selection are shown in Table 4. A middle ground design was eventually utilized. This design uses an external IM6402 Intersil UART and generates the clock for transmitter and receiver operation on one of the NSC810 chips. The Intersil IM6402 UART data sheet [Ref. 7] contains specific information on chip operation. A system block and pin diagram are provided in

TABLE 4. UART PORT ASSIGNMENTS

Address (Hex)	Read (R)/ Write (W)	Assignment
A0		UART Data
A1 - BF		*** NOT USED ***
C0		UART Status
C1 - CF		*** NOT USED ***

Figures 16 and 17.

A number of features make the selection of Intersil's UART feasible for the controller. The first is its CMOS construction. Secondly, the Intersil UART is compatible with the industry standard for IM6402. This allows it to be easily connected to any number of available "dumb" terminals or microcomputers. The UART also has the capability of being programmed in a number of different data formats. This programmable feature allows it to be interfaced in a number of different serial data environments. The following discussion, on the UART's operation, will be broken up into six areas: receiver operation, transmitter operation, status, control, setup, and clock specifications.

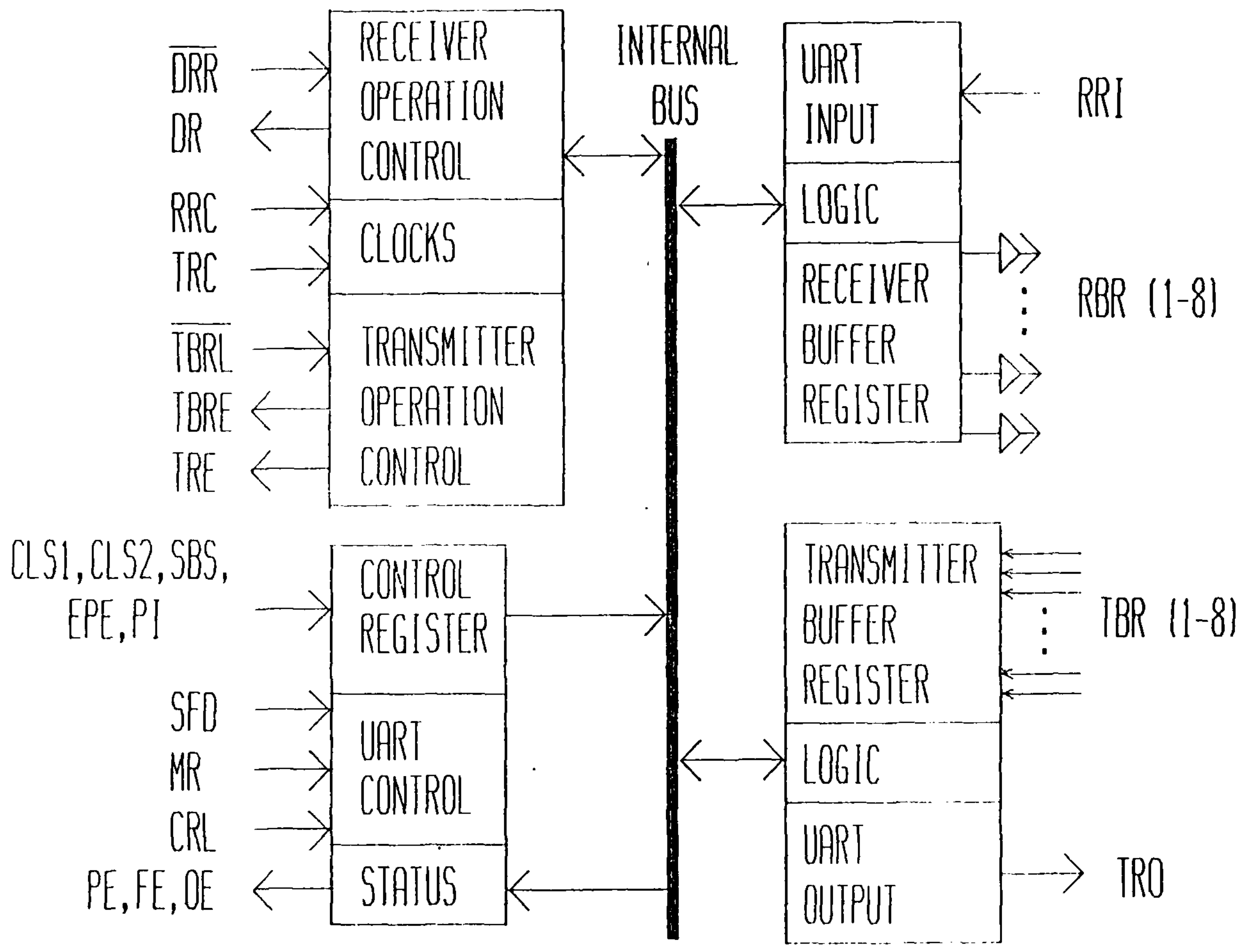


Figure 16. UART Block Diagram

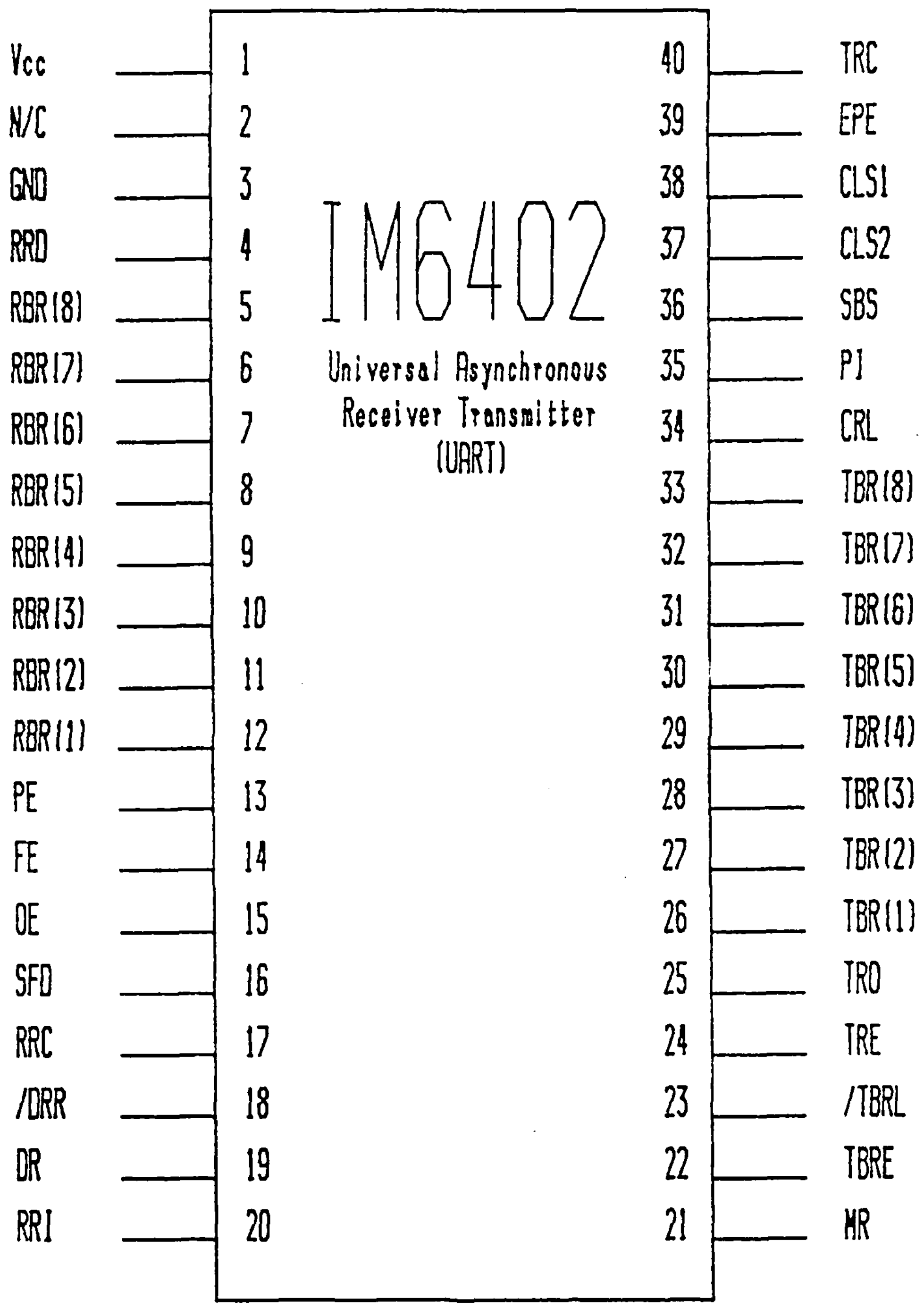


Figure 17. UART Chip Pinout

In receiver operation, external asynchronous serial data is brought into the UART, its format is changed, and then the message is sent over the controller's parallel data bus to the NSC800. Following is a description of receiver operation at the pin level. Pin 4 on the UART is the Receiver Register Disabled (RRD) line. This line controls the Receiver Register Holding Register outputs by bringing them to a high impedance state. Serial data enters the UART through the Receiver Register Input (RRI), pin 20. After the data enters, it is sent to the Receiver Buffer Register (RBR) on the UART. A signal on the Data Received (DR) line, pin 19, indicates that the data is in the buffer.

The UART's transmitter takes data in from the controller's parallel data bus, changes its format, and then transmits the message out. There are a number of lines on the UART dedicated to transmit operations. The following discussion will look at most of them. An indication, that the transmitter buffer is ready for new data is provided on the Transmitter Buffer Register Ready (TBRE) line, pin 22. The Transmitter Buffer Register Load (/TBRL) line, pin 23, is used to bring parallel data into the UART. This data is brought into the Transmit Buffer Registers (TBR), pins 26-33. Data is eventually transmitted out of the Transmitter Register Output (TRO), pin 25. The Transmitter Register Empty (TRE) line, pin 24, gives an indication that data transmission is complete.

The user interface is greatly simplified by status indications available on Intersil's UART. Besides the transmit and receive status indications, other status signals include the Parity Error (PE), Framing Error (FE), and Overrun Error (OE). Parity error is an indication that the received message has incorrect bits. Either the parity bit or another bit in the transmission is in error. If the first stop bit in the transmission is in error a framing error occurs. The overrun error indicates that a data received indication has not been received before the last character was sent to the receiver buffer registers.

Three control signals the UART uses are Status Flag Disable (SFD), Master Reset (MR), and Control Register Load (CRL). The Status Flag Disable places all status outputs in a high impedance state, thus removing them from the bus. The Control Register Load is used to load the system's particular operating environment. This environment or setup information will be described later. Finally, every time the UART is powered up, the Master Reset is used to return the UART to a known state for programming purposes.

The UART can be used in wide variety of formats. These formats are displayed in Table 5 [Ref.8]. The actual setup utilized depends on the choise of Parity Inhibit (PI), Stop Bit Select (SBS), Character Select Length (CLS1 and CLS2) and Even Parity Enable (EPE). The Parity Inhibit line is used to stop parity checking when a data byte is received

TABLE 5. UART DATA FORMATS

CONTROL WORD							
CLS2	CLS1	PI	EPE	SBS	DATA BITS	PARITY BIT	STOP BIT(S)
L	L	L	L	L	5	ODD	1
L	L	L	L	H	5	ODD	1.5
L	L	L	H	L	5	EVEN	1
L	L	L	H	H	5	EVEN	1.5
L	L	H	X	L	5	DISABLED	1
L	L	H	X	H	5	DISABLED	1.5
L	H	L	L	L	6	ODD	1
L	H	L	L	H	6	ODD	2
L	H	L	H	L	6	EVEN	1
L	H	L	H	H	6	EVEN	2
L	H	H	X	L	6	DISABLED	1
L	H	H	X	H	6	DISABLED	2
H	L	L	L	L	7	ODD	1
H	L	L	L	H	7	ODD	2
H	L	L	H	L	7	EVEN	1
H	L	L	H	H	7	EVEN	2
H	L	H	X	L	7	DISABLED	1
H	L	H	X	H	7	DISABLED	2
H	H	L	L	L	8	ODD	1
H	H	L	L	H	8	ODD	2
H	H	L	H	L	8	EVEN	1
H	H	L	H	H	8	EVEN	2
H	H	H	X	L	8	DISABLED	1
H	H	H	X	H	8	DISABLED	2

and parity generation when transmitting a word. If it is desirable to check parity, the Even Parity Enable line is used to check either odd or even parity. The Stop Bit Select and Character Length Select lines allow the UART to interface to a variety of transmitted word lengths and formats.

The clock signal which the UART uses, comes from one of the controller's NSC810s. This signal is 16 times the desired transmit and receive rate. The UART has the capability of operating at baud rates in excess of 200K if provided with a clocking signal in the range of 4 MHz. This particular system is designed for 9600 baud. The required clocking for this rate is a little over 150 KHz. A baud rate of 9600 is selected because it supports the "dumb" terminal used with the system. This rate also supports a variety of industry standard "dumb" terminal emulation communication packages.

The data path between the UART and "dumb" terminal has to be determined. The industry standard RS-232-C 25 pin interconnection is used to interface the two units. The line drivers shown in Figure 18 are used on the UART's serial transmit output and receive input. The connection uses a three wire cable and is shown in Figure 19. The three lines are Transmitted Data, Received Data, and Signal Ground.

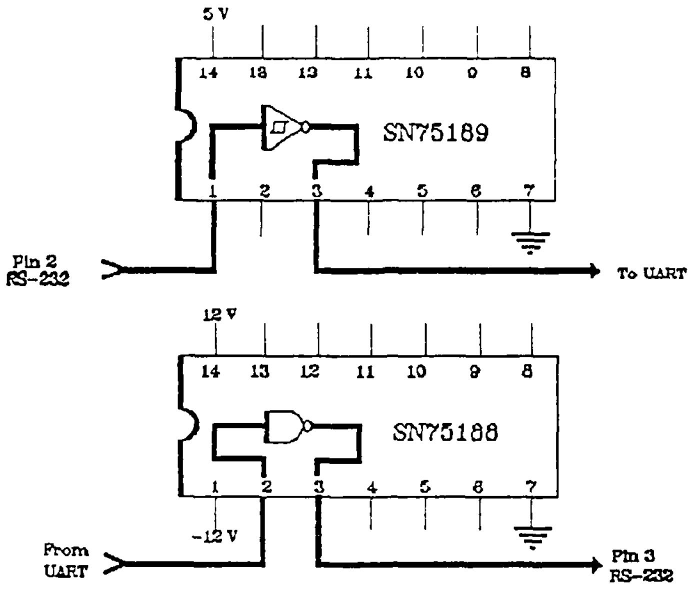


Figure 18. UART Transmit and Receive Line Drivers

E. REAL TIME CLOCK

With a requirement to initiate experiments at a particular time, some type of microprocessor compatible real time clock is required. The National Semiconductor MM58167A Microprocessor Real Time Clock is used. Specific information on the real time clock is provided in the associated data sheet [Ref. 9]. As with most of the other components on the controller, it is CMOS in construction. Some other

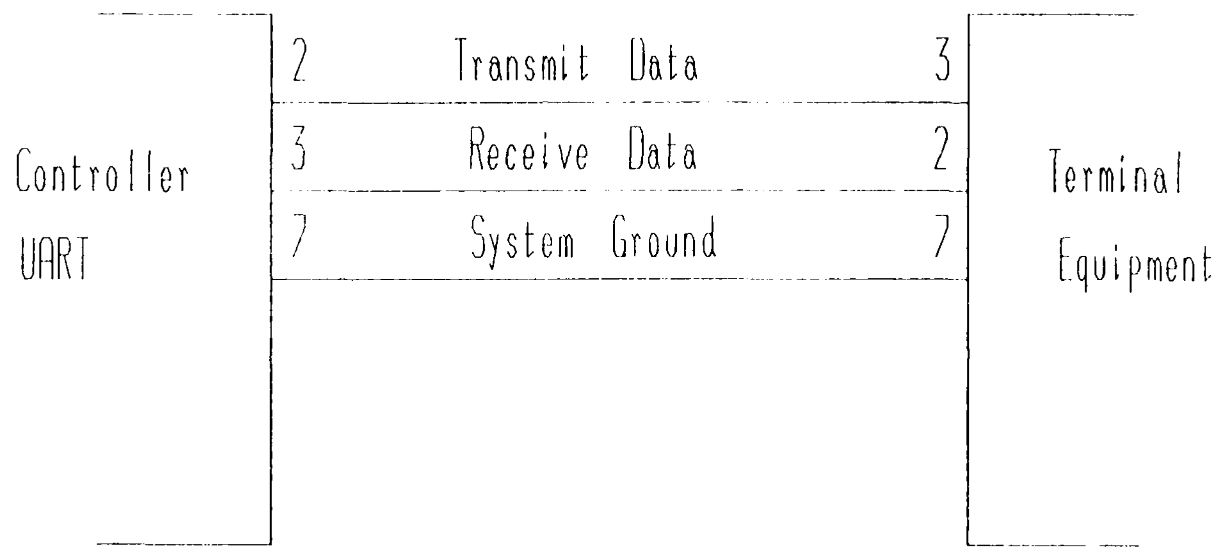


Figure 19. UART-to-Dumb Terminal RS-232 Interface

clock features are presented in the following discussion. Month to thousandths of a second information is provided by this clock. With a local four year calendar, the clock can be used for very long term applications. A power-down feature allows it to be disabled from the rest of the system for low power operations. The clock counter is divided into two 4-bit Binary Coded Decimal (BCD) digits. During each read and write operation the two digits can be accessed. The BCD real time clock format is provided in Table 6 [Ref. 10]. The chip has an alarm clock feature which can be preprogrammed. This is needed for timed experiment initiation. The clock can also be programmed to give a periodic signal output with its variable interrupt control features. The variety of available programmable functions are seen on the port diagram in Table 7. There is a status check available to ensure that clock rollover has not occurred during a real time fetch.

A block and pin diagram are provided in Figures 20 and 21 respectively. These diagrams, along with the following discussion, should clarify clock operation. Along with the standard lines, such as Chip Select (/CS), Power (Vdd), Ground (Vss), and Read (/RD) and Write (/WR) are some lines unique to the clock's operation. The first of which is the Ready (RDY) line, pin 4, which indicates that the data requested is now valid to be read. The Oscillator Input (OSC

TABLE 6. BCD REAL TIME CLOCK FORMATS

COUNTER ADDRESSED	UNITS				MAX USED BCD CODE	TENS				MAX USED BCD CODE
	D0	D1	D2	D3		D4	D5	D6	D7	
Ten Thousandths of a Second	0	0	0	0	0	I/O	I/O	I/O	I/O	9
Tenths and Hundredths of Seconds	I/O	I/O	I/O	I/O	9	I/O	I/O	I/O	I/O	9
Seconds	I/O	I/O	I/O	I/O	9	I/O	I/O	I/O	0	5
Minutes	I/O	I/O	I/O	I/O	9	I/O	I/O	I/O	0	5
Hours	I/O	I/O	I/O	I/O	9	I/O	I/O	0	0	2
Day of the Week	I/O	I/O	I/O	0	7	0	0	0	0	0
Day of the Month	I/O	I/O	I/O	I/O	9	I/O	I/O	0	0	3
Month	I/O	I/O	I/O	I/O	9	I/O	0	0	0	1

TABLE 7. REAL TIME CLOCK PORT ASSIGNMENT

Address (Hex)	Read (R)/ Write (W)	Assignment
60		Counter - Ten thousandths of a second
61		Counter - Hundreths and tenths of a second
62		Counter - Seconds
63		Counter - Minutes
64		Counter - Hours
65		Counter - Day of week
66		Counter - Day of month
67		Counter - Month
68		RAM - Ten thousandths of a second
69		RAM - Hundreths and tenths of a second
6A		RAM - Seconds
6B		RAM - Minutes
6C		RAM - Hours
6D		RAM - Day of Week
6E		RAM - Day of Month
6F		Ram - Months
70		Interrupt Status Register
71		Interrupt Control Register
72		Counter Reset
73		RAM Reset
74		Status Bit
75		Go Command
76		Standby Interrupt
77 - 7E		*** NOT USED ***
7F		Test Mode

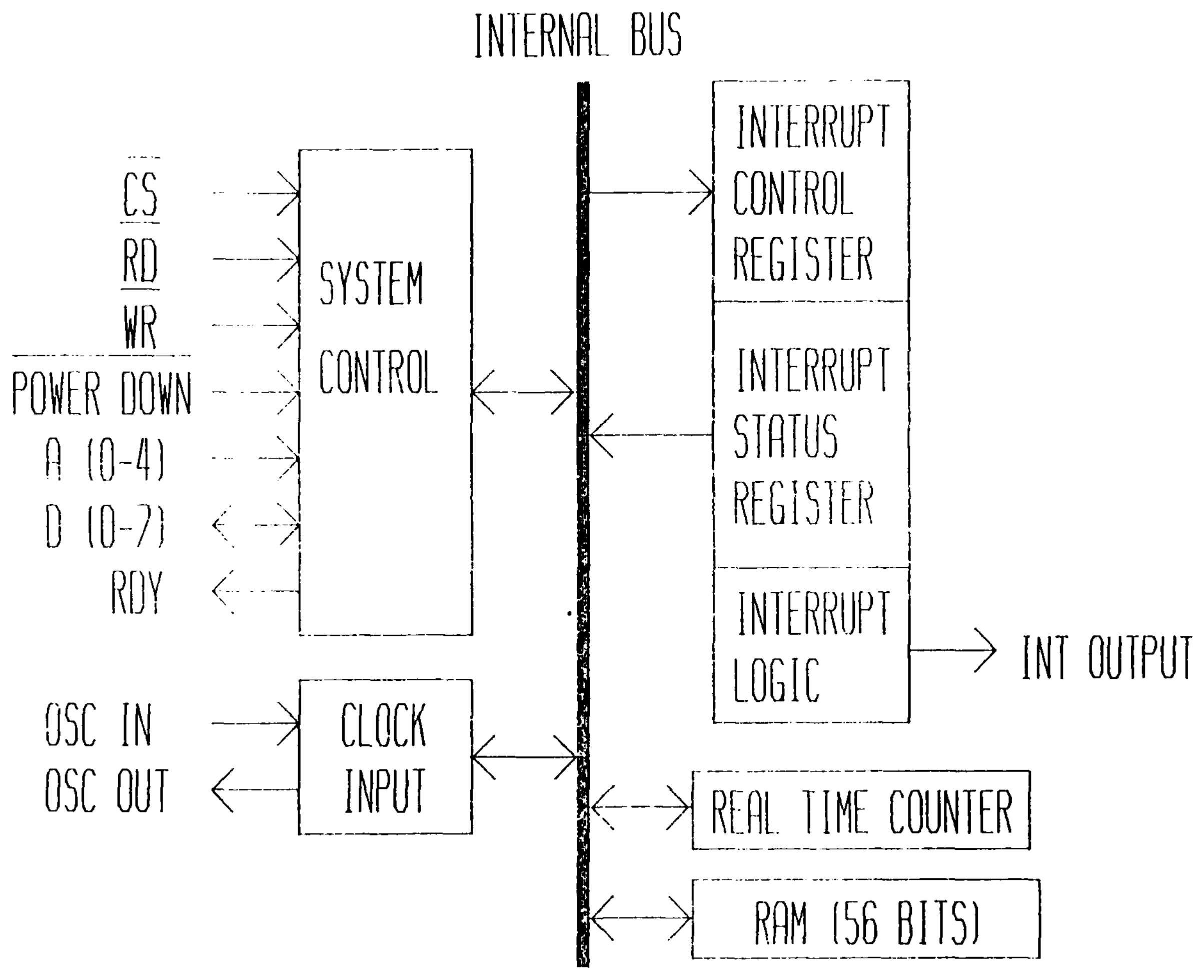


Figure 20. Real Time Clock Block Diagram

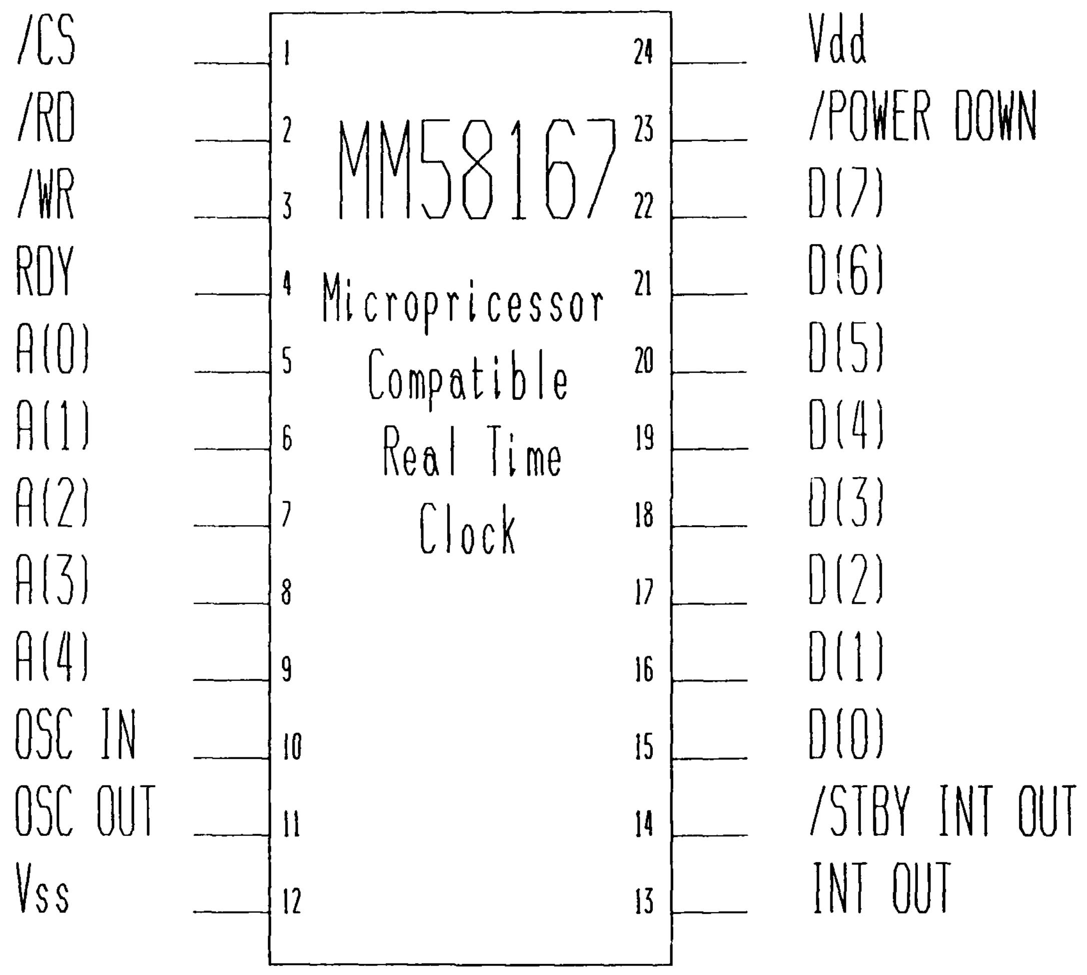


Figure 21. Real Time Clock Chip Pinout

IN), pin 10, and its accompanying output (OSC OUT), pin 11, are used in the crystal oscillator circuitry shown in Figure 22. An Interrupt Output (INT OUT) at pin 13 is programmed to provide eight different intervals of output. The final pin unique to the real time clock is the Standby Interrupt (/STBY INT OUT) line. This is the only line enabled during low power operations. All other outputs are driven to a high impedance state during low power operations.

F. STATIC RAM

The Hitachi HM6116P-2 2048-word x 8-bit High Speed Static CMOS RAM is chosen for two reasons. First, it can be integrated into the controller system very easily because no refresh requirements have to be addressed. Secondly, it supports very low power operation. If dynamic RAM is chosen, the package size for a certain amount of memory will be smaller, but the power dissipation incurred will be greater than if static RAM is chosen. The controller will have 8K of local static RAM. It will be used as a buffer space while accumulating enough data for a bubble memory write. It will also be used as working space for the controller's NSC800. The associated pin diagram of the HM6116P-2 is provided in Figure 23. A truth table of RAM read and write operations is provided in Figure 24. Specific information on RAM operation is provided in the Hitachi IC memory data sheet [Ref. 11].

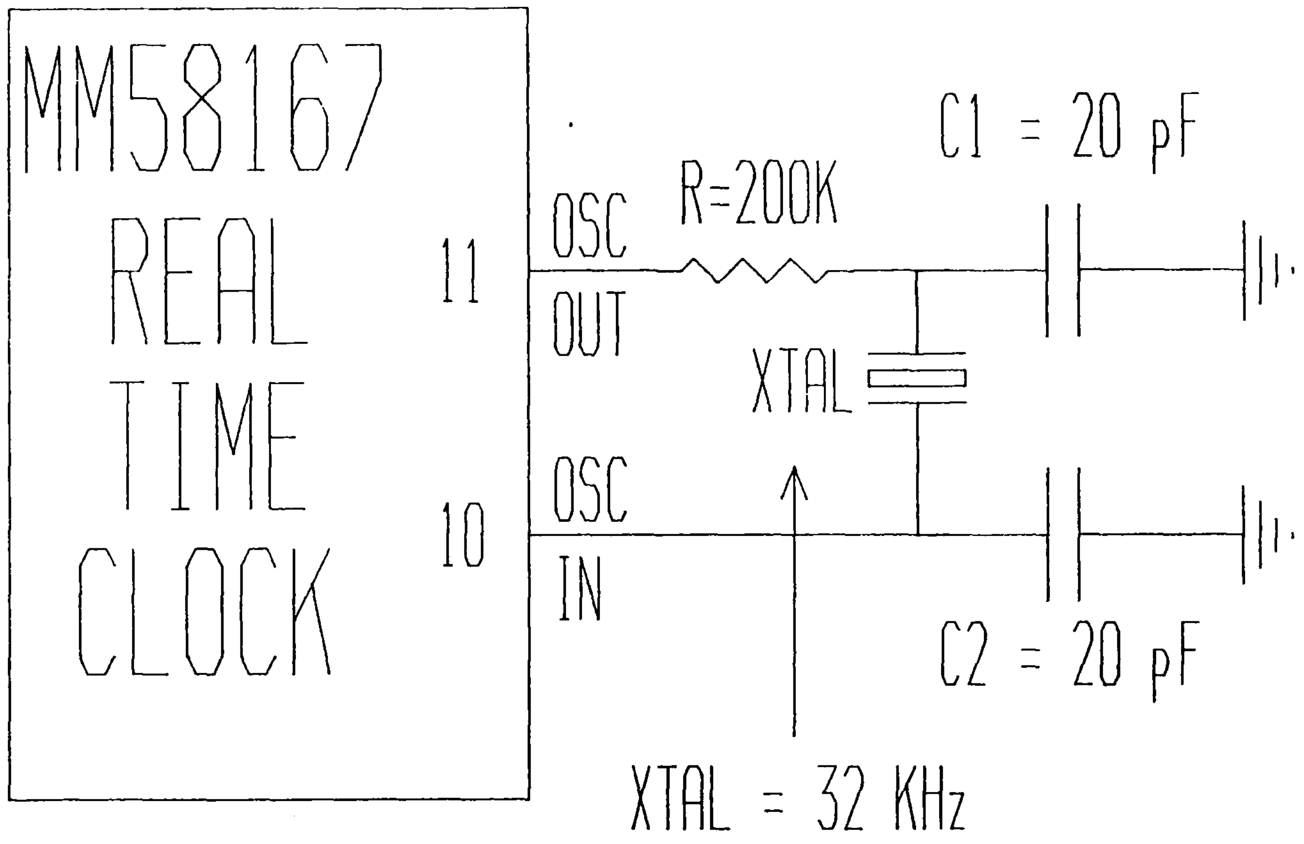


Figure 22. Real Time Clock Oscillator Circuitry

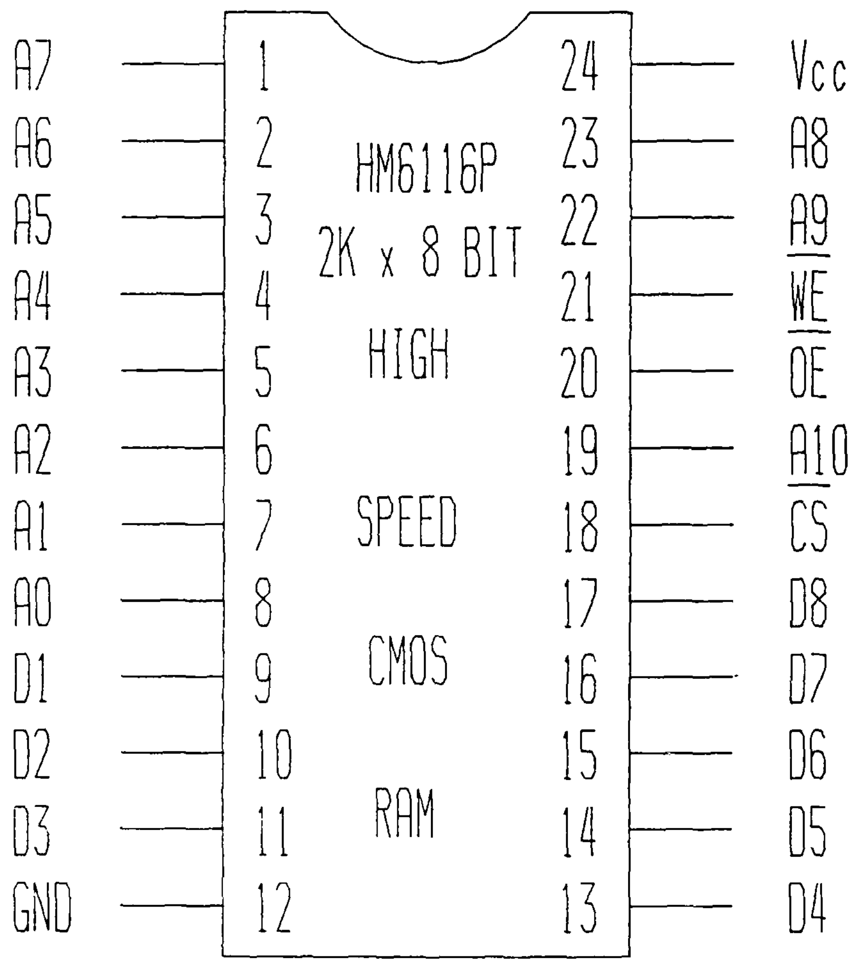


Figure 23. Static RAM Chip Pinout

MODE	$\overline{CS}$	$\overline{OE}$	$\overline{WE}$
READ	L	L	H
WRITE (1)	L	H	L
WRITE (2)	L	L	L
NOT SELECTED	H	x	x

Figure 24. Static Ram Truth Table

G. EPROM UTILIZATION

The drivers for operation are resident in four Intel 2732A 32K (4Kx8) UV Erasable PROMs. Specific details on the EPROM are provided in the associated data sheet [Ref. 12]. EPROMs are chosen over ROMs primarily because program development time is shorter and development costs are lower. The 12K bytes of EPROM memory is considered an adequate amount for resident memory requirements. The pin diagram of the EPROM is provided in Figure 25. The Intel 2732A also features a low power standby mode.

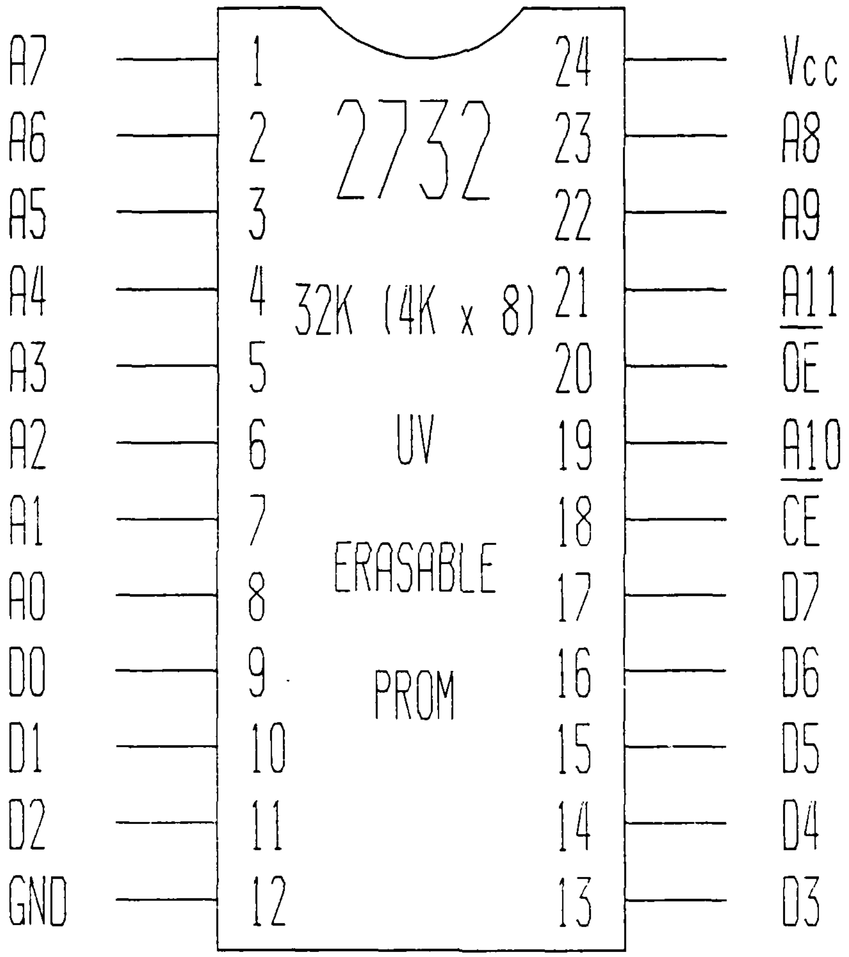


Figure 25. EPROM Chip Pinout

III. LANGUAGE SELECTION

A number of factors have to be evaluated before the programming language is selected. Some of the considerations that must be addressed are longevity of the project, extended I/O control, real-time control, and memory space availability. The question of longevity, short or long term use, must be answered to determine if the language will require much maintenance and growth during its lifetime. A short term project can be written in a very inflexible manner, and overall performance will not be overly affected. On the other hand, if the duration of the project is over a number of years a great amount of flexibility and growth potential must be built in. The environment in which the controller will be operating is fairly well defined and short term. The controller, as was earlier stated, will have a great deal of I/O control. The experiments the controller will be controlling fall into two general categories: systems with no local logic and systems with local logic and handshaking capability. With a well defined operating environment to work in and a project driven with sequential real time control, controller programming can be fairly rigid and well contained. This well defined environment can be resident in a small amount of memory. While a high level language is well suited for programming in a general purpose environment, it tends to have large memory requirements. On the other hand, very specific tasking and a great deal of machine level control, make an assembly level approach more feasible. This approach is appropriate if the code size is not excessively large. In assembly language programs, the code tends to be more optimized, further reducing memory usage. It was against this backdrop that the Z-80 and 8080 assembly languages were chosen for controller drivers. All system drivers are written at the assembly level. In the future, overall system control will be written in a compiled higher level language. A majority of programming is done in Z80 rather than 8080 assembly because it has a number of enhanced features. Some of these included code that performs more functions than similar 8080 code and more diverse I/O instructions.

IV. CONTROLLER SYSTEM DESIGN

Now that the components that make up the system have been presented, the way in which they work together will be developed next. A short recapitulation of the requirements that the overall design should satisfy will be presented. Small size and low power dissipation are physical constraints. The small size necessitates that a small compact design be developed. The requirement of low power dictates that low power chips be utilized.

The controller is almost 100 percent CMOS in construction. There are also some additional benefits associated with the use of CMOS devices. One being a higher degree of noise immunity than TTL circuitry. This is evident in the varying power environment in which the CMOS chips can operate. Another advantage of CMOS is in its ability to be common-bussed. This bussing arrangement is allowed because three-state transmission gates are provided on most CMOS devices. With almost all CMOS components the problem of compatibility among chips is minimized. Overall system loading is reduced due to the CMOS design and the requirement for bus drivers is minimized. Although there are a number of benefits associated with CMOS construction, some precautions associated with their use have to be looked at. Since unstable operation may be observed if an unused input to the

CMOS device is left open, it should be tied to either a high or low logic level depending on circuit requirements. The main power dissipator in the controller, the bubble memory storage device, is the primary non-CMOS device present. When the bubble device is not in the actual process of reading or writing it is completely powered down. The primary trade off that this introduces is a 160 millisecond delay at most before the bubble can be written to again. The requirement for extensive port control brings about the inclusion of three NSC810s with their many ports and timer capabilities. A block diagram of the controller's primary component interconnections and their control, data and address bus is provided in Figure 26. The direction of data flow on the controller's data bus is controlled by an octal bus transceiver. A consolidated controller I/O map is provided in Figure 27. The decoder shown in Figure 28 is used to select the various controller components. The controller's decoder and NSC800 provide signals that are used to determine directional flow on the data bus. This circuitry is shown in Figure 29. A total of 8K bytes of RAM and 12K bytes of EPROM will be utilized for controller operation. The controller memory map is provided in Figure 30. The memory decode circuitry is shown in Figure 31.

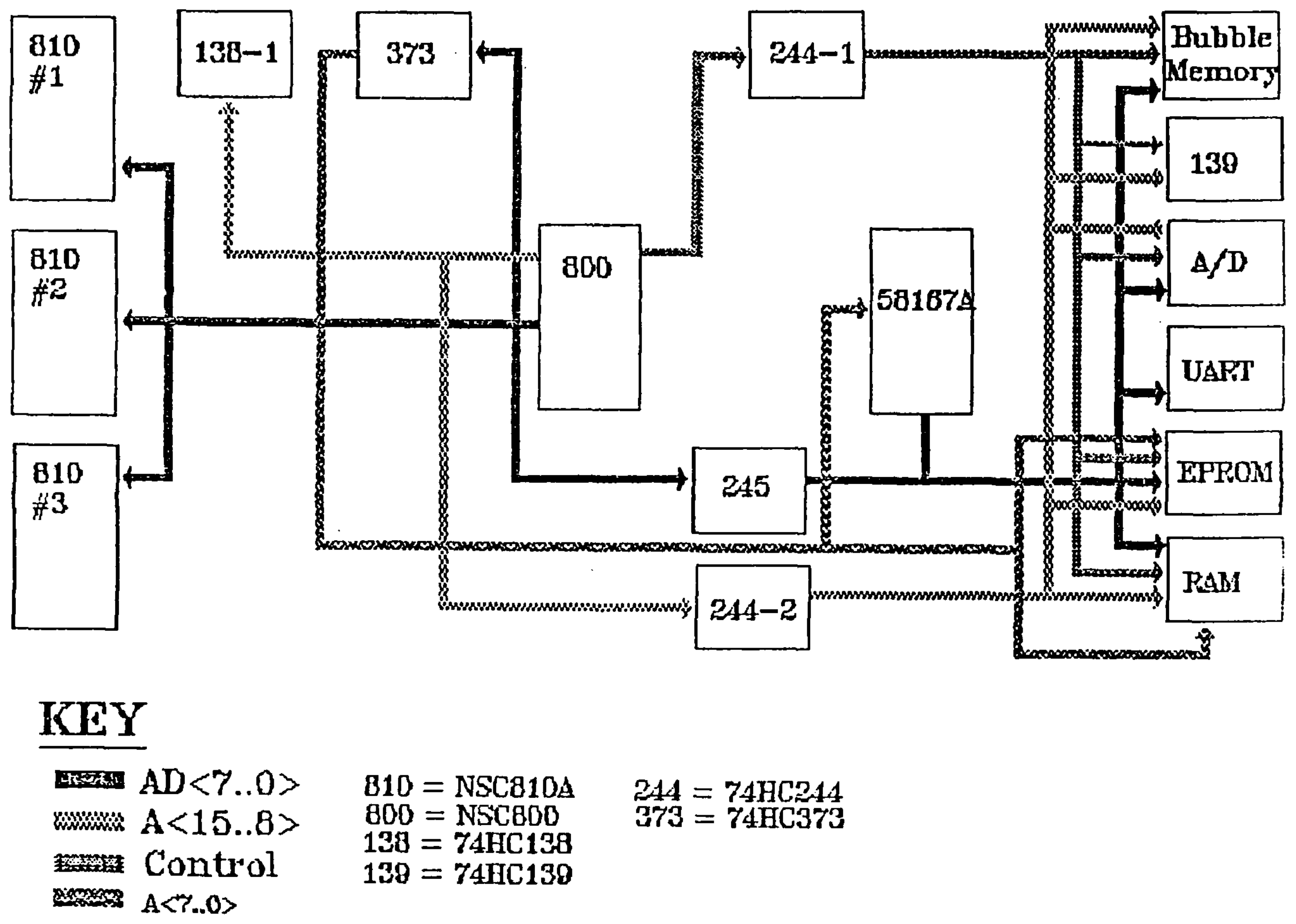


Figure 26. Controller Data, Address, and Control Paths

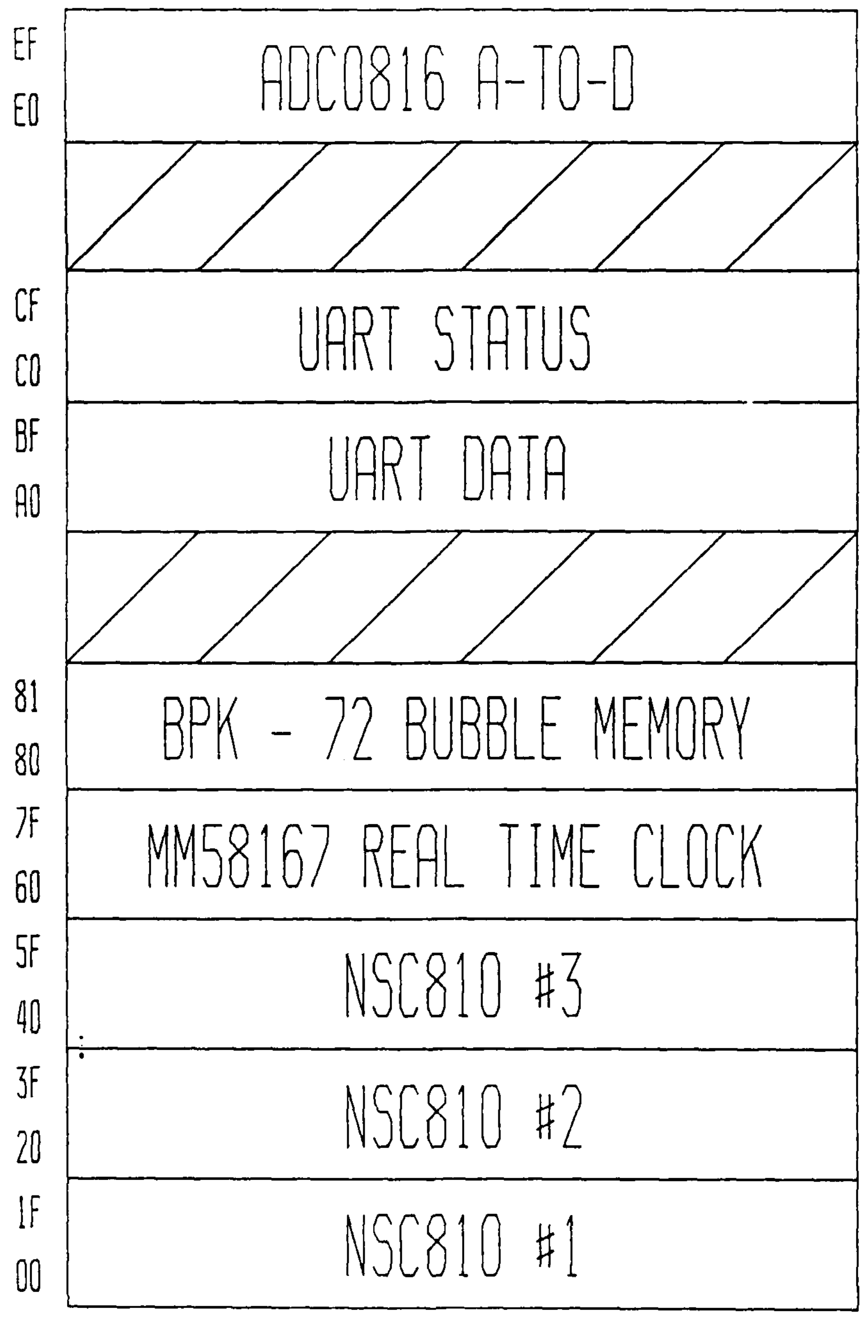


Figure 27. Controller I/O Map

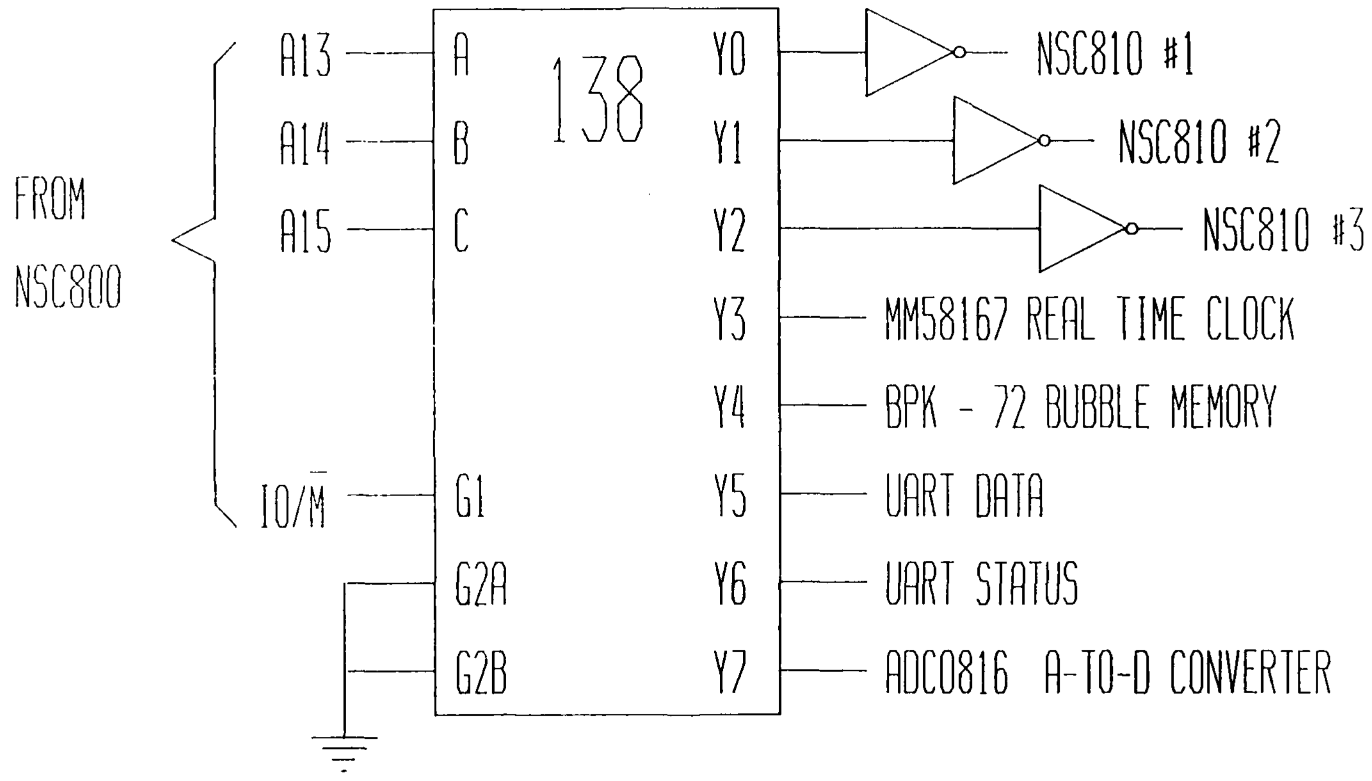


Figure 28. Controller Component Decoder

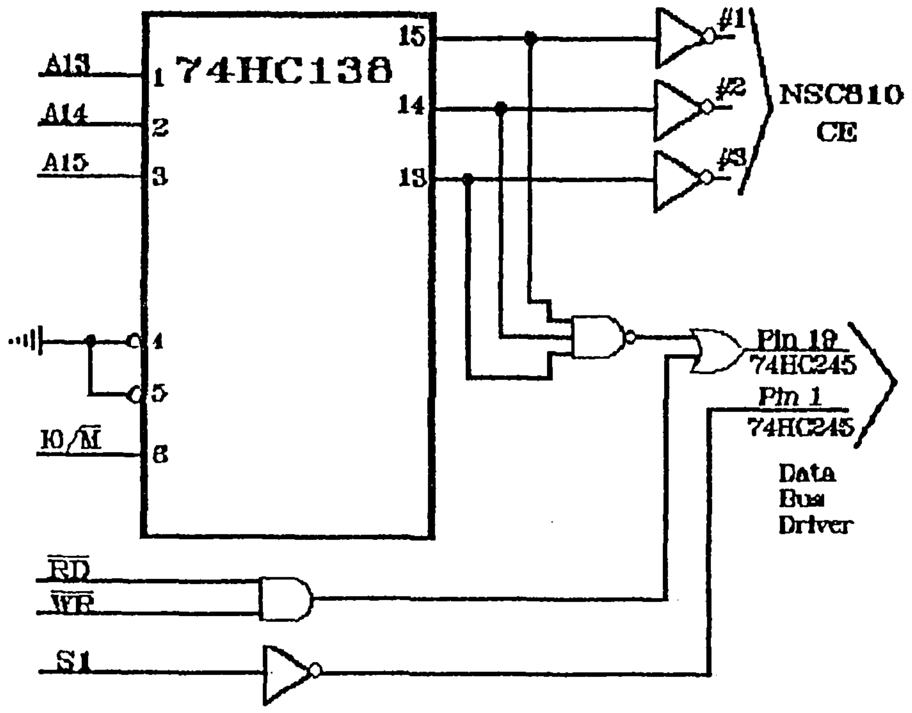


Figure 29. Data Bus Direction Circuitry

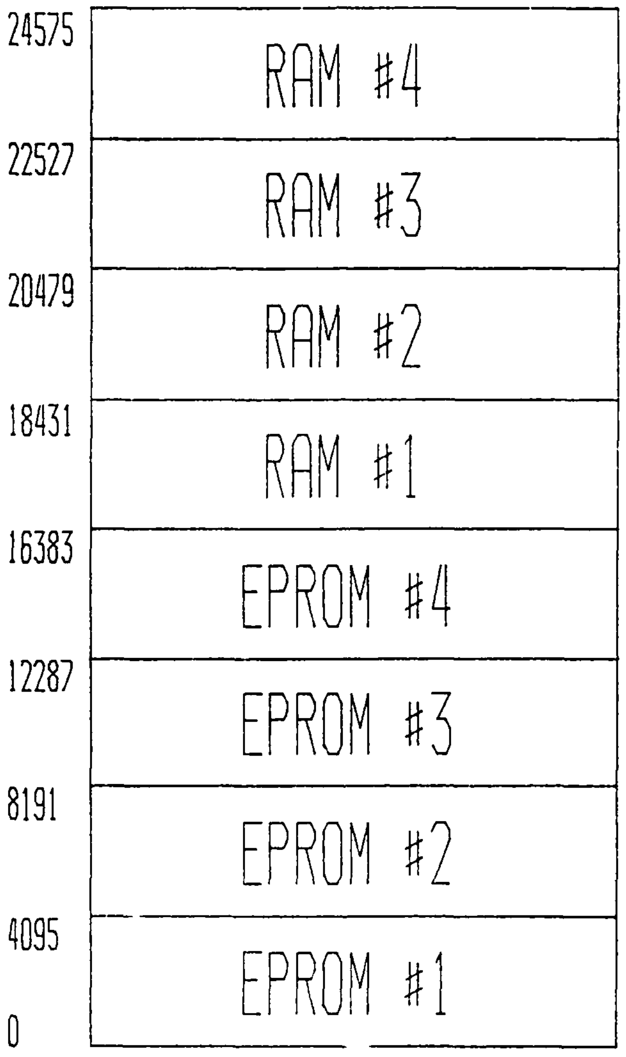


Figure 30. Controller Memory Map

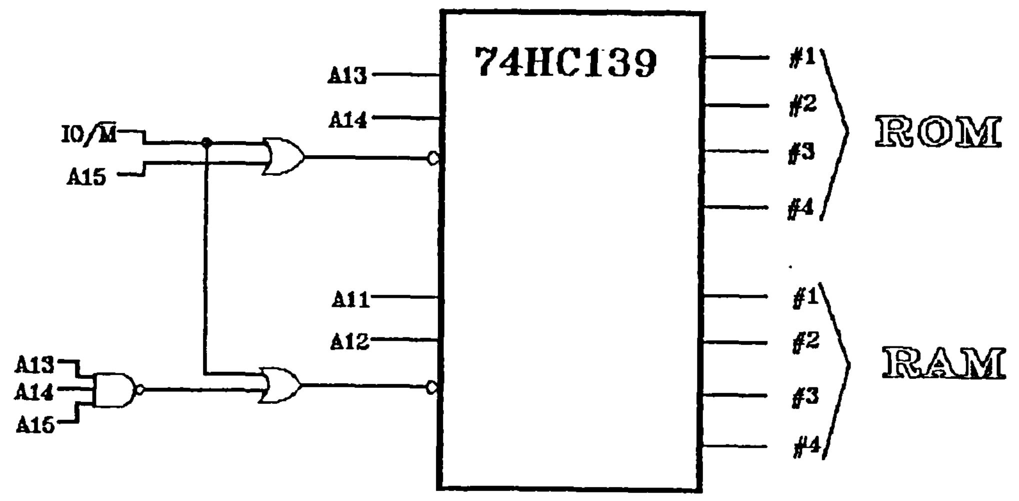


Figure 31. Memory Decode Circuitry

V. GENERAL SYSTEM LAYOUT AND CONSTRUCTION

After ensuring that system components will satisfy controller requirements, their interconnection is made on paper. This procedure takes a number of steps. First an overall and general view of system's interconnections is made. Next comes the more specific task of designing a system layout with pin identification. The controller system layout is provided in Appendix A. Through the layout, the intricacies of component interaction can be observed. One can also observe the underlying outline of the circuit's ultimate construction. The analog-to-digital converter's circuit interface with the controller is shown in Figure 32.

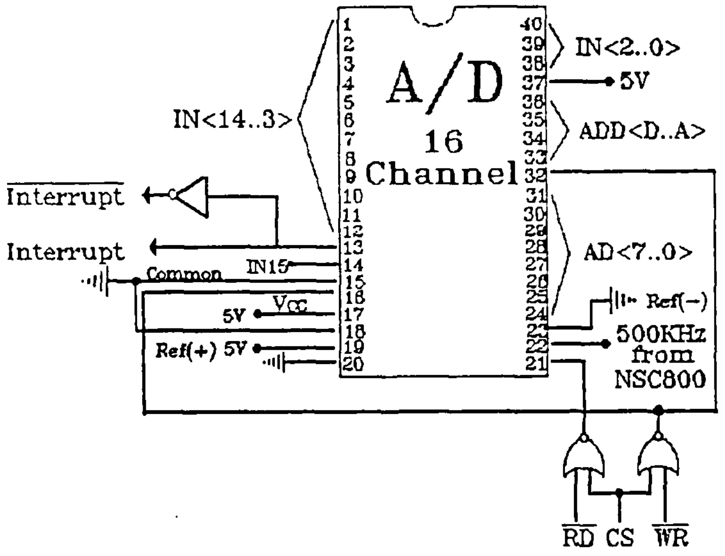


Figure 32. Controller Interface with the A-to-D Converter

Following the paper design the circuit is realized in hardware. Specific details, such as chip socket positioning, bus layout, and the utilization of bypass capacitors to remove spikes in the system have to be addressed. A brief description of the building techniques used in the controller's construction follows.

The first situation that has to be evaluated is whether the prototype circuit will be created by a wirewrap or solderless breadboard assembly. Due to the frequency that the controller will be operating, around 4 MHz, the wirewrap option is chosen. This will avoid possible ringing problems associated with breadboards operating at high frequencies. After choosing the wirewrap approach, the actual layout of the components has to be made. An evaluation of the data, address, and control lines among the various chips is made and through layout these path lengths are minimized as much as possible. Space for the RS-232 and other interconnections with external equipment is also identified. The controller wirewrap board layout is provided in Figure 33. The ground and power lines are first routed. In wiring power, the leads are of heavy gauge and as short as possible. The effectiveness of a power supply can be seriously degraded by the resistance in its lead lines between power source and load. The ground lines are also of heavy gauge. Throughout the circuit, .01 microfarad bypass capacitors are used to remove any unwanted current spikes that might be caused by

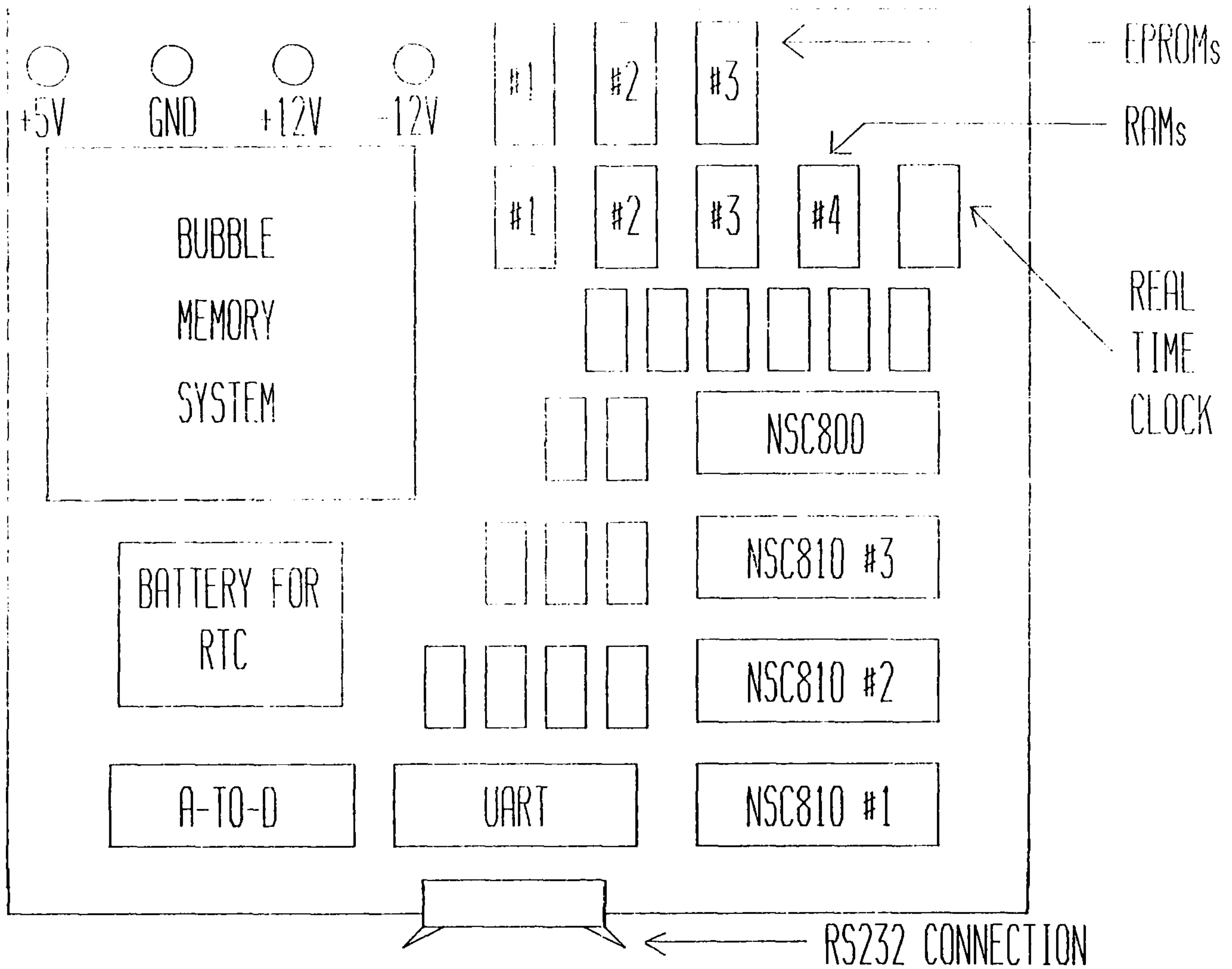


Figure 33. Controller Wirewrap Board Layout

circuit switching action from totem-pole devices. After wiring is complete, a check of the power and ground leads is made by a continuity test to all applicable chip connections. Next the external power supply is connected to the circuit and all power and ground lines are again checked. In the prototype controller, power is provided through an external power supply. The required decay rates that are needed for proper bubble operation are satisfied. Next the address/data, low address, and high address busses are wirewrapped and checked via a continuity test of all lines. The system control lines are connected and are also checked with a continuity test. The NSC800, a NSC810, a EPROM and one RAM are placed on the controller. The interaction between the NSC800 and EPROM is tested by a short program. This program causes a fetch, the jump instruction is decoded and then two more bytes of memory are fetched. This results in a toggling of the control and memory select lines. This is verified with an oscilloscope. The remaining RAM is added to the circuit along with the UART. The UART is then verified by a short program that takes keyboard inputs, sends them to the controller, and then back to the terminal screen. Now the bubble memory module is added to the controller. Its operation is then verified. The procedure used is provided in following chapters. The real time clock and analog-to-digital converters are then added to the controller.

VI. SOFTWARE FLOWCHART AND DRIVER DEVELOPMENT

A. BUBBLE MEMORY CONTROL

After hardware development comes its application. While some components interface quite easily with the microprocessor and only need to be addressed to return an answer, others require a series of operations or drivers to function. The NSC800 to bubble memory system interface is the most complicated one on the controller. The following discussion will look into what is required to initialize the bubble system, read from it, and write to it. A bubble command summary is provided in Table 8 [Ref. 13]. This should help in the following driver development. The drivers and the main program calling them are provided in Appendix B. Information on bubble memory programming from the _BPK 72 Bubble Memory Prototype Kit User's Manual_ [Ref. 14] and a bubble-to-controller software interface [Ref. 15] are used in driver development.

The interface between the bubble and NSC800 microprocessor is heavily dependent on software. The drivers control all aspects of bubble operation. Data is received from the NSC800 microprocessor and is converted into bubble memory commands or command execution parameters. The driver also interprets signals from the bubble indicating that a

TABLE 8. BUBBLE MEMORY COMMAND SUMMARY

D3	D2	D1	D0	Command
0	0	0	0	Write Bootloop Register Masked
0	0	0	1	Initialize
0	0	1	0	Read Bubble Data
0	0	1	1	Write Bubble Data
0	1	0	0	Read Seek
0	1	0	1	Read Bootloop Register
0	1	1	0	Write Bootloop Register
0	1	1	1	Write Bootloop
1	0	0	0	Read FSA Status
1	0	0	1	Abort
1	0	1	0	Write Seek
1	0	1	1	Read Bootloop
1	1	0	0	Read Corrected Data
1	1	0	1	Reset FIFO
1	1	1	0	MBM Purge
1	1	1	1	Software Reset

particular operation has been completed or that it has failed. This information is then returned to the NSC800. Bubble driver utilization is multilayered. The upper layer comprises the main program and user interface. A lower layer is divided into a number of specific operations. These operations then call primitive subroutine actions. Some of these primitives include, initializing parame ric registers, resetting the FIFO data buffer, writing to ubble memory, and reading bubble data memory. The bubble system has the capability of operating in a polled, DMA or interrupt driven mode. The polled mode is used for this controller application. There are a couple of reasons for this selection. Because the bubble is used to store historical data during the experiment, the rate at which the bubble will be written to and read from is very low. A polled mode more than adequately satisfies controller operation. With no additional hardware required to initiate polled operation, its interface with the microprocessor will be easier. The primary disadvantage associated with polled operation is the excessive time the microprocessor is tied up with the bubble. Data transfer is software controlled. After a command is sent to the bubble, the status register is read continuously. This register determines when data is to be written to or read from the bubble FIFO. All bubble operation is composed of smaller driver primitives. Some of these primitives include abort, purge, FIFO reset and the read/write commands. Typical polled operation command execution and data transfer routine flowcharts are provided in Figures 34 [Ref. 16] and 35 [Ref. 17]. All commands sent to the bubble utilize similar input and output formats. The main difference is the actual commands sent. Operation is divided into initializing, writing to, or reading from the bubble memory system. Bubble memory initialization is carried out each time it is powered up. In the process of initialization, the following bubble commands are performed: abort, purge, FIFO reset, and read/write to bootloop. In write operations the parametric registers are first written to. This establishes the size of page, that will be written and where in the bubble memory it will be placed. The bubble memory's data write operation is then initiated. Data is taken from a predefined location in RAM memory to the bubble memory system. With the read command, data from the bubble memory is written to a preselected RAM buffer area. Like the write operation, the parametric registers establish the type of transfer that will occur. The actual read bubble data command is then issued.

B. REAL TIME CLOCK

The four operations that the real time clock performs are setting time, programming the interrupt, reading time, and setting the alarm output. To set the time, its associated function is called and the user is prompted for the

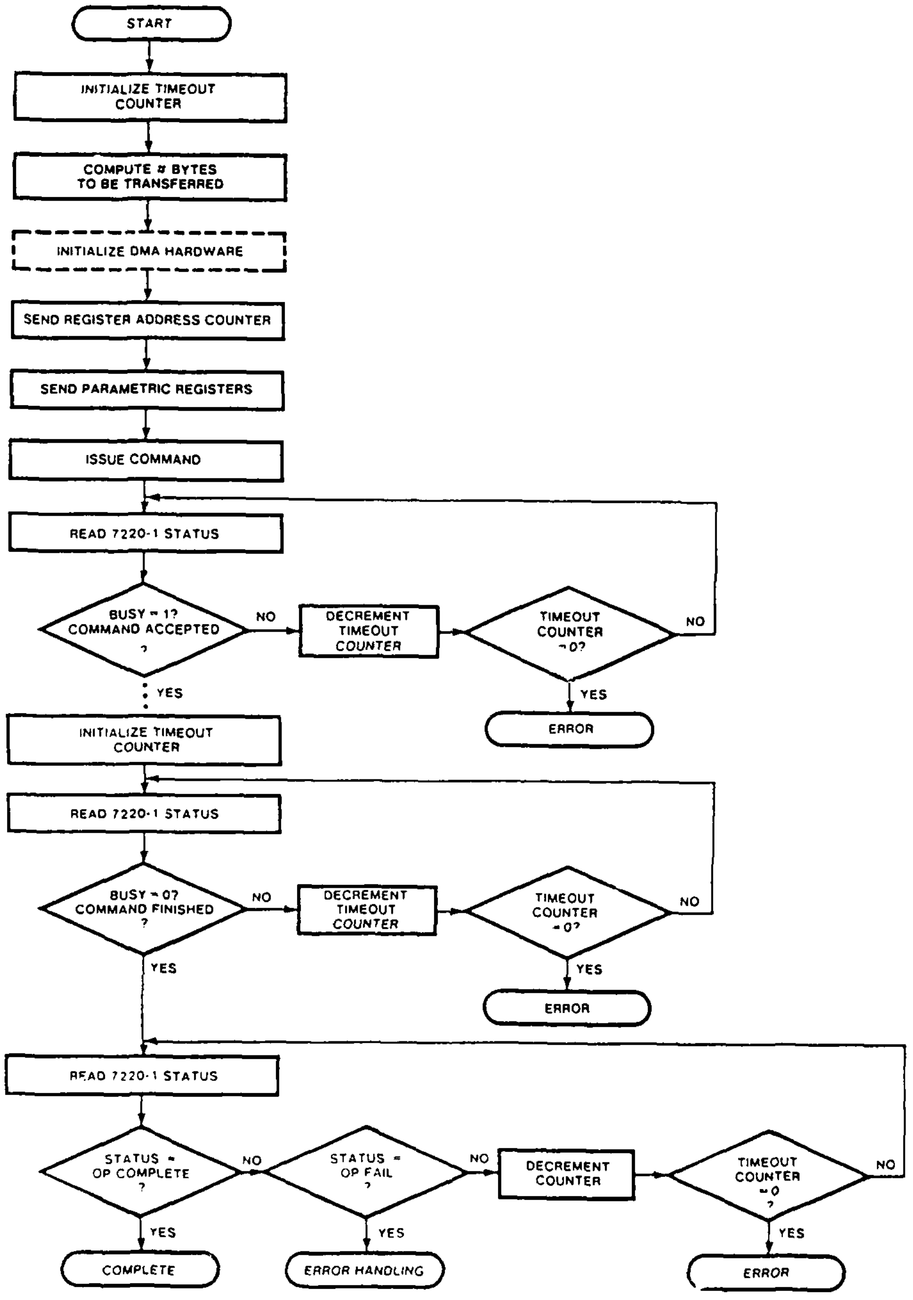


Figure 34. Polled Operation Command Execution Flowchart

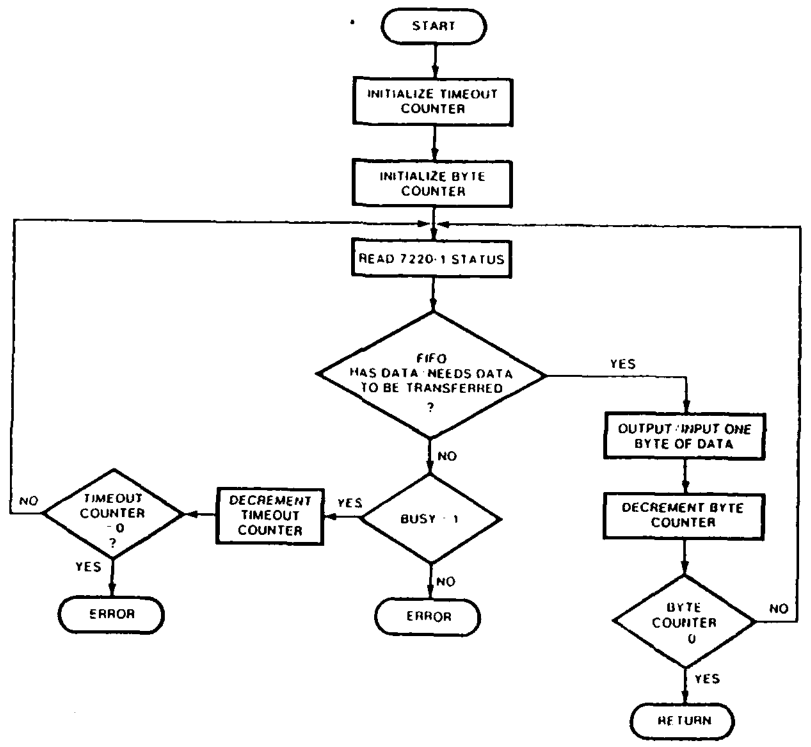


Figure 35. Polled Operation Data Transfer Flowchart

appropriate inputs. These inputs are converted to BCD format and are sent to the appropriate port of the clock. The alarm type function, that the real time clock performs, is similar to the clock set operation. Rather than write to the clock's counter as it does in the time set operation, the alarm is written to the clock's latches or RAM. The clock's interrupt output is maskable through the interrupt control register and can be programmed to any of eight possible signals. The interrupt control register regulates which of the bits in the interrupt status register has an output. The interrupt status register has interrupt outputs for tenths of seconds, seconds, minutes, hours, week, day of month, and month. The interrupt control register is written to by calling the interrupt function. After this function is called there is a prompt on the terminal for the desired interrupt cycle. A byte with the desired interrupt bit set to a "1" is then sent to the interrupt control register. The interrupt register format is provided in Figure 36. To read the present time, the program's read time function is invoked. After determining the time, the clock's status bit is checked to see if the clock rolled over during the read. If no roll over occurs the present time is returned. The real time clock also has a power down feature which is invoked by bringing the power down line to a logical 0. The standby interrupt is the only output allowed during power

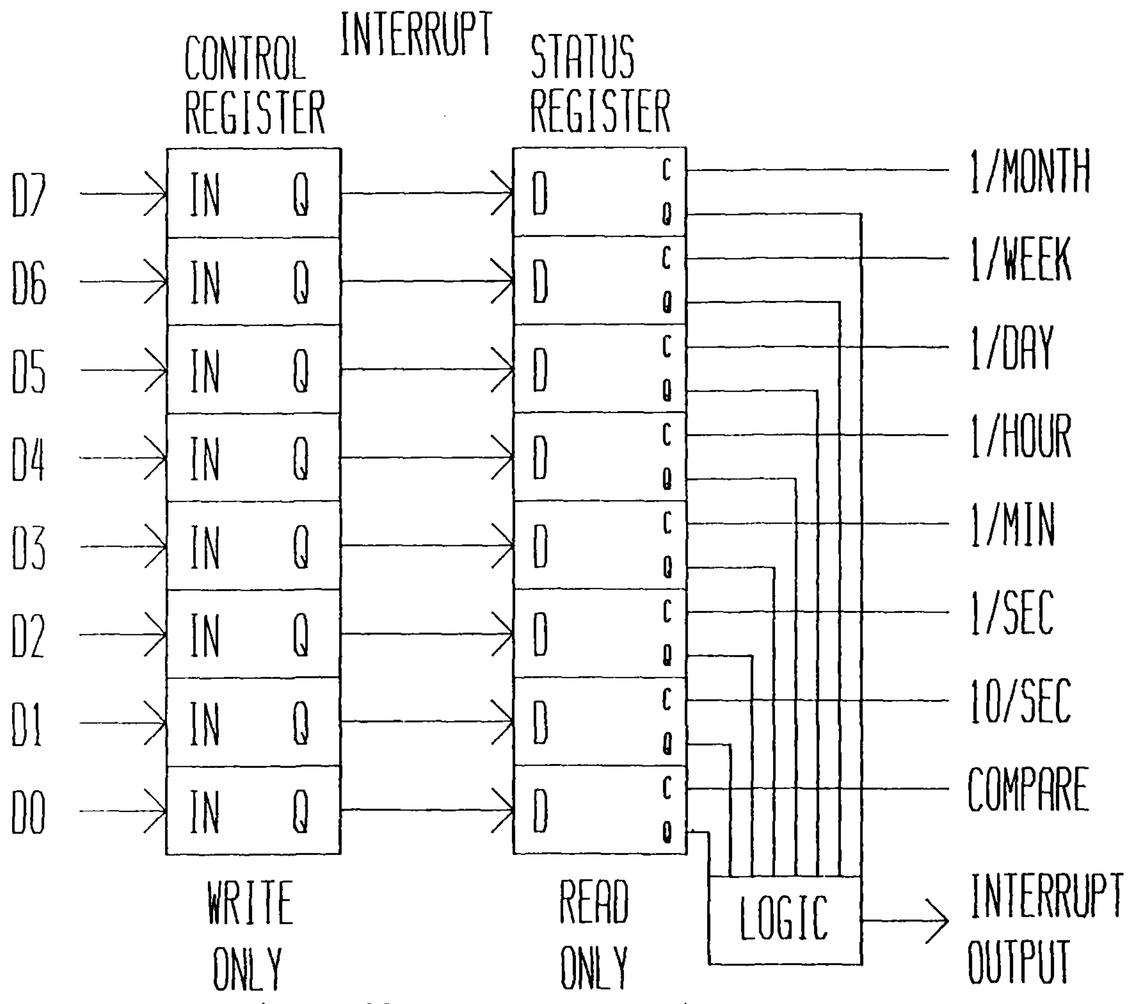


Figure 36. Interrupt Register Format

down and is enabled by writing a 1 on the D0 line when the standby interrupt is selected.

C. CONFIGURING THE I/O PORTS

The port characteristics of the controller are established on the NSC810s. To aid in the following discussion, the three NSC810 port diagrams are shown in Tables 9, 10 and 11. The ports of the NSC810 can be configured in a number of modes as previously discussed. The A port byte can be configured as basic I/O, strobed mode input, strobed mode output, or strobed mode output with tri-state. The configuration or mode is selected by writing to the mode definition register. The allowable mode definition register assignments and the bytes that select them are provided in Figure 37. The B and C ports are *only configured* as basic I/O. The C port is also utilized as a programmable timer output or serves as a handshake register when the A port is in the strobed mode. After the port mode is selected, the direction of the port is determined. This selection is made by writing to the Data Direction Register of the port. A "0" at a particular bit location indicates the bit is configured as an input. A "1" in a bit location signifies an output. When the A port is configured in one of the strobed modes the registers of the A and C ports require the configurations shown in Figure 38. The individual bits of the A, B and C ports can be manipulated with their respective bit-set

TABLE 9. NSC810A NO.1 PORT ASSIGNMENT

Address (Hex)	Read (R)/ Write (W)	Assignment
00	R/W	PA 0-7 (Ext. General Purpose Strobed Input)
01	R/W	PB 0-4 (UART Control)
		PB 5-7 (Available)
02	R/W	PC 0-2 (Power Control)
		PC 3-5 (A/D Counter Timer)
03		*** NOT USED ***
04	W	DDR - Port A
05	W	DDR - Port B
06	W	DDR - Port C
07	W	Mode Def Reg
08	W	Port A (Bit - Clear)
09	W	Port B (Bit - Clear)
0A	W	Port C (Bit - Clear)
0B		*** NOT USED ***
0C	W	Port A (Bit - Set)
0D	W	Port B (Bit - Set)
0E	W	Port C (Bit - Set)
0F		*** NOT USED ***
10		Timer 0 (LB) UART Baud Rate
11		Timer 0 (HB) UART Baud Rate
12		Timer 1 (LB) A/D Clock
13		Timer 1 (HB) A/D Clock
14	W	Timer 0 Stop
15	W	Timer 0 Start
16	W	Timer 1 Stop
17	W	Timer 1 Start
18	R/W	Timer Mode (0)
19	R/W	Timer Mode (1)
1A - 1F		*** NOT USED ***

TABLE 10. NSC810A NO.2 PORT ASSIGNMENT

Address (Hex)	Read (R)/ Write (W)	Assignment
20	R/W	PA 0-7 (SSDR Strobed Input)
21	R/W	PB 0-7 (Power Group Input)
22	R/W	PC 0-2 (SSDR Control)
		PC 3-5 (Available)
23		*** NOT USED ***
24	W	DDR - Port A
25	W	DDR - Port B
26	W	DDR - Port C
27	W	Mode Def Reg
28	W	Port A (Bit - Clear)
29	W	Port B (Bit - Clear)
2A	W	Port C (Bit - Clear)
2B		*** NOT USED ***
2C	W	Port A (Bit - Set)
2D	W	Port B (Bit - Set)
2E	W	Port C (Bit - Set)
2F		*** NOT USED ***
30		Timer 0 (LB) Power Group
31		Timer 0 (HB) Power Group
32		Timer 1 (LB) Available
33		Timer 1 (HB) Available
34	W	Timer 0 Stop
35	W	Timer 0 Start
36	W	Timer 1 Stop
37	W	Timer 1 Start
38	R/W	Timer Mode (0)
39	R/W	Timer Mode (1)
3A - 3F		*** NOT USED ***

TABLE 11. NSC810A NO.3 PORT ASSIGNMENT

Address (Hex)	Read (R)/ Write (W)	Assignment
40	R/W	PA 0-7 (SSDR Strobed Output
41	R/W	PB 0-7 (Power Group Output
42	R/W	PC 0-2 (SSDR Output)
		PC 3-5 (Available)
43		*** NOT USED ***
44	W	DDR - Port A
45	W	DDR - Port B
46	W	DDR - Port C
47	W	Mode Def Reg
48	W	Port A (Bit - Clear)
49	W	Port B (Bit - Clear)
4A	W	Port C (Bit - Clear)
4B		*** NOT USED ***
4C	W	Port A (Bit - Set)
4D	W	Port B (Bit - Set)
4E	W	Port C (Bit - Set)
4F		*** NOT USED ***
50		Timer 0 (LB) Power Group
51		Timer 0 (HB) Power Group
52		Timer 1 (LB) Available
53		Timer 1 (HB) Available
54	W	Timer 0 Stop
55	W	Timer 0 Start
56	W	Timer 1 Stop
57	W	Timer 1 Start
58	R/W	Timer Mode (0)
59	R/W	Timer Mode (1)
5A - 5F		*** NOT USED ***

MODE	DESCRIPTION	MODE DEFINITION REGISTER BIT ASSIGNMENTS							
		7	6	5	4	3	2	1	0
0	BASIC I/O	x	x	x	x	x	x	x	0
1	STROBED MODE INPUT	x	x	x	x	x	x	0	1
2	STROBED MODE OUTPUT (ACTIVE)	x	x	x	x	x	0	1	1
3	STROBED MODE OUTPUT (TRI-STATE)	x	x	x	x	x	1	1	1

Figure 37. Mode Definition Register Byte Assignment

MODE	MDR	PORT A DDR	PORT C DDR	PORT C LATCH
STROBED OUTPUT (ACTIVE)	xxxxx011	11111111	xxx011	xxx1xx
STROBED OUTPUT (TRI-STATE)	xxxxx111	11111111	xxx011	xxx1xx
STROBED INPUT	xxxxxx01	00000000	xxx011	xxx1xx

Figure 38. A and C Port Strobed Mode Configuration

and clear ports. If a "0" is written to the port, no change will occur. If a "1" is written to a particular bit location, the selected operation will occur.

The procedure to set the clock output of the NSC810 follows. First, the timer being configured is stopped by writing a 000 or 111 in the timer mode register. Next the desired clock mode is written to the timer mode register. The six timer modes available are presented in Figure 39. The modulo value for the clock is then written into the modulus register, low byte and then high byte. The selected clock is then started. The mode 5 configuration, square wave clock, is chosen for UART and A/D timing signals. An example of the square wave output and the effect the modulo value has on the signal is provided in Figure 40.

D. ANALOG-TO-DIGITAL CONVERTER

For a read on the analog-to-digital converter, the particular channel is selected, the microprocessor initiates a delay and then the channel voltage is read.

E. UART

A description of the drivers for the read and write operations follow. For a keyboard read, the status register of the UART is activated and a check is made to see if a character has been received and transferred to the receiver buffer register. After the character is received, it is

		TMR		
MODE	DESCRIPTION	D0	D1	D2
0	TIMER STOP / RESET	0	0	0
1	EVENT COUNTER	1	0	0
2	ACCUMULATIVE TIMER	0	1	0
3	RESTARTABLE TIMER	1	1	0
4	ONE SHOT	0	0	1
5	SQUARE WAVE	1	0	1
6	PULSE GENERATOR	0	1	1
7	TIMER STOP / RESET	1	1	1

Figure 39. Real Time Clock Timer Modes

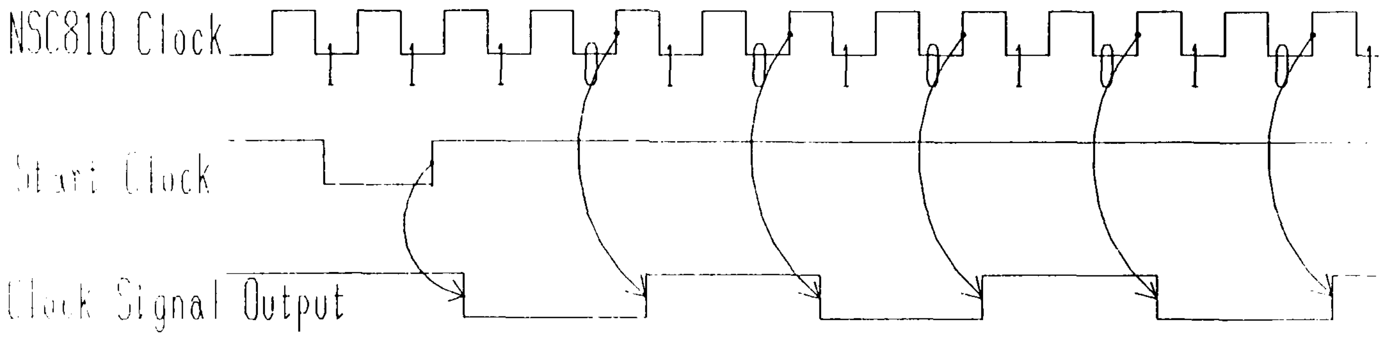
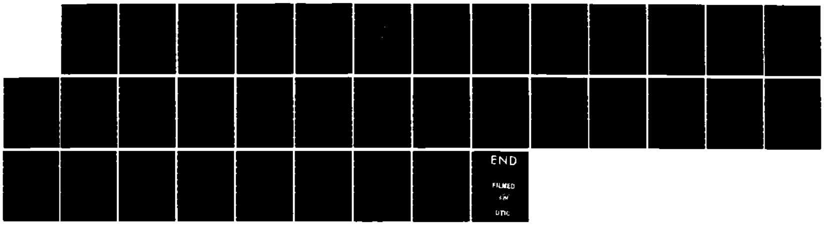
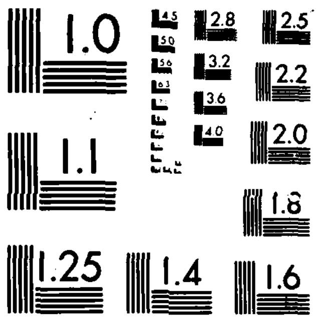


Figure 40. NSC810A Square Wave Output

sent to the microprocessor. For transmissions out to the console, the status register is again read. This time, the transmitter buffer register empty indication is checked. After an indication that data has been transmitted to the transmitter register and the UART is ready for new data, an output from the NSC800 is made.


END
FILMED
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
45
50
56
63

VII. CONCLUSIONS

The controller designed and built in this thesis satisfies the requirements of the space shuttle experiment. Although designed for a specific experiment, it has the flexibility to be applied to following space shuttle projects. There is the possibility for hardware and software enhancements.

A. FUTURE HARDWARE AND SOFTWARE POSSIBILITIES

The controller's EPROM memory can be increased from its present 12K bytes to 56K bytes. This would yield an overall RAM and EPROM memory usage of 64K bytes, the maximum allowable in this controller configuration. The bubble memory unit on the controller can be upgraded to the newer Intel 4M bit system, DMA data circuitry could provide greater data throughput, and interrupt data transfer circuitry can decrease program overhead. Digital-to-analog converter circuitry can also be incorporated and a range of voltages would then be available for system control. Finally, a newer 16 or 32 bit CMOS microprocessor can replace existing microprocessor control devices.

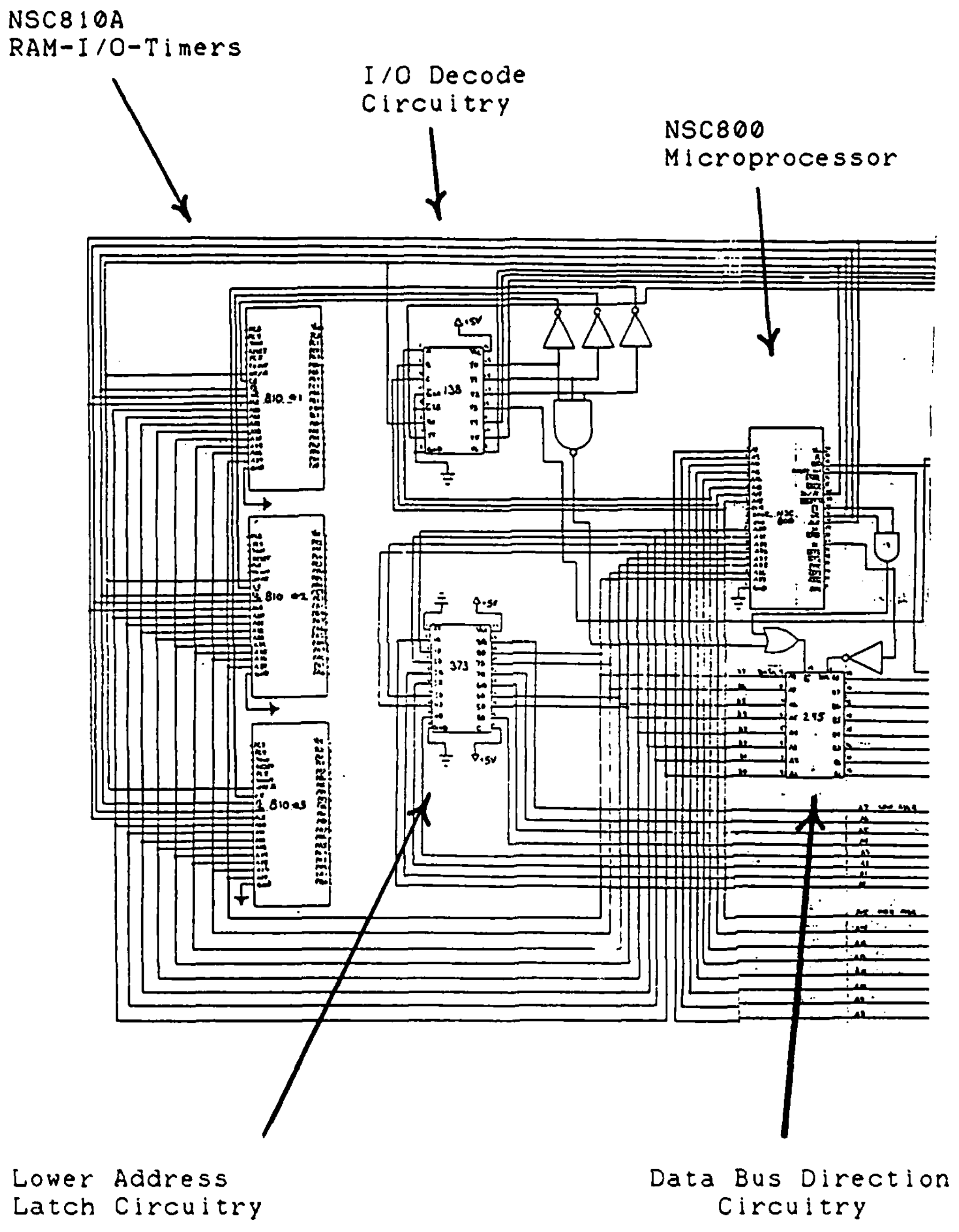
An extensive real-time operating system can be developed for the controller. System programming during this thesis was exclusively EPROM based.

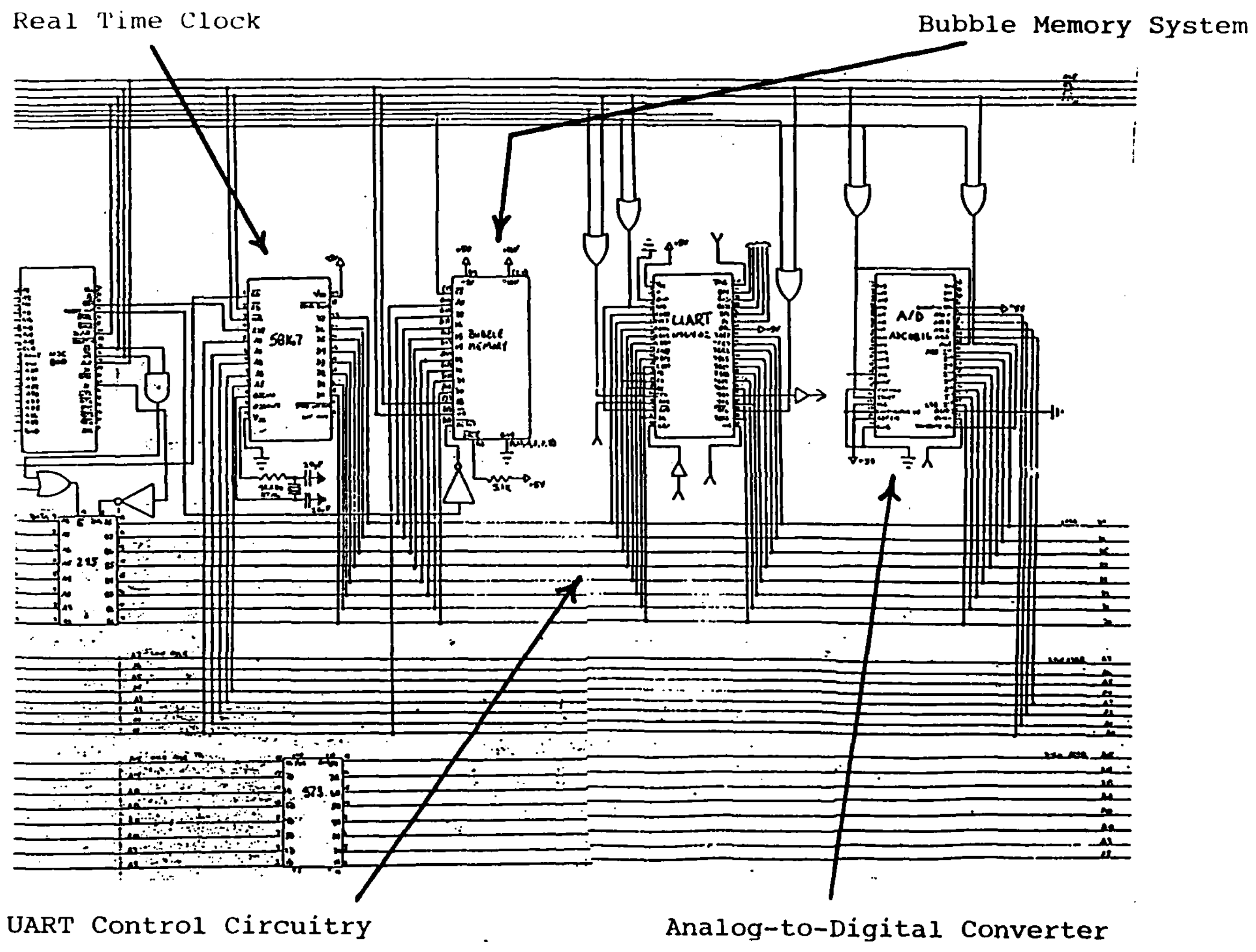
B. FUTURE APPLICATIONS AND RESEARCH OPPORTUNITIES

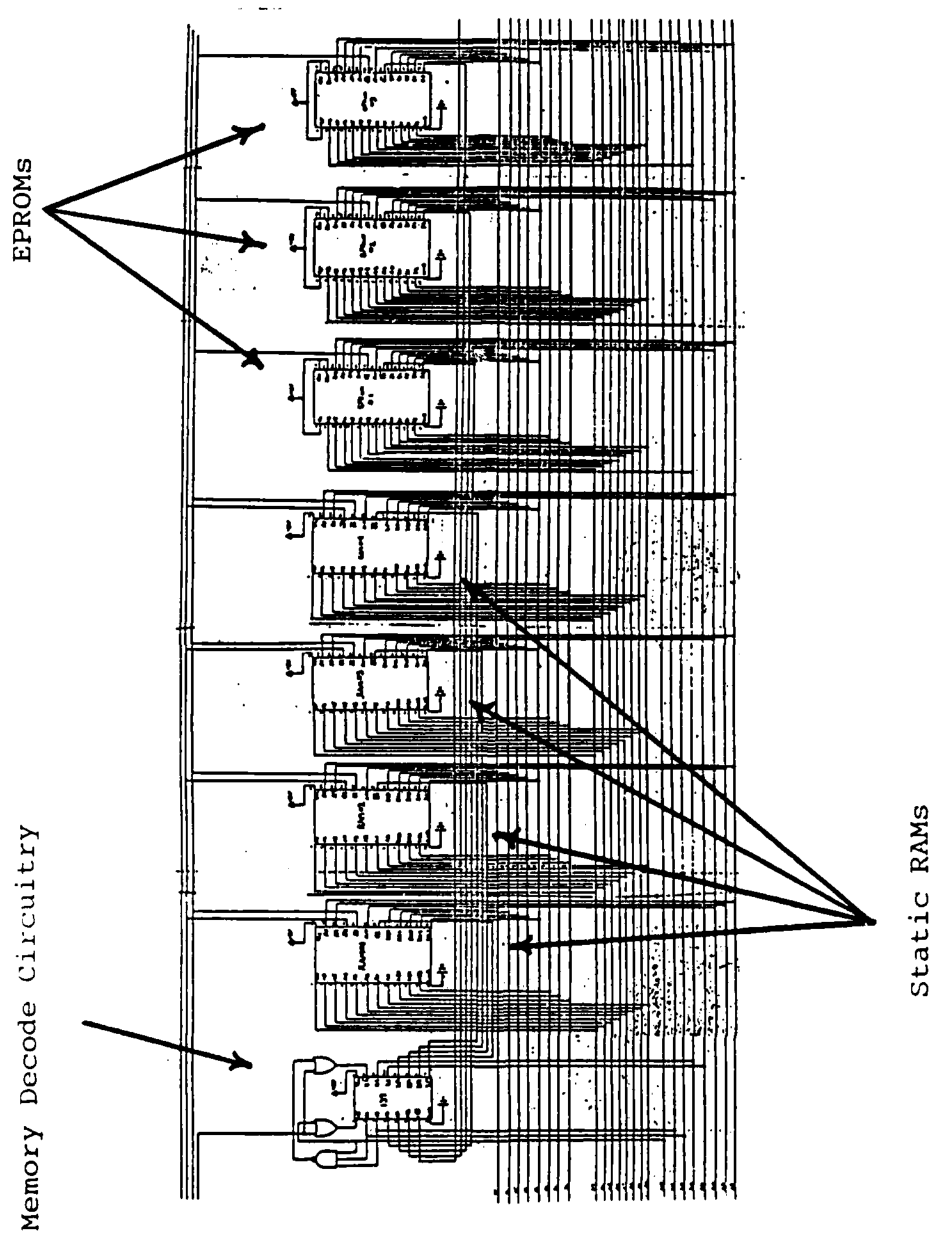
Hardware and software enhancements previously discussed contain ideas that can be used in future controller research and development. The controller can also function as a low-cost general purpose control workstation for the classroom and can support many different control system environments. The UART allows it to be interfaced with most microcomputers.

APPENDIX A

CONTROLLER CIRCUIT LAYOUT




Real Time Clock
Bubble Memory System
58167
BUBBLE MEMORY
UART
A/D
ADC0816
273
573
UART Control Circuitry
Analog-to-Digital Converter


EPROMs
Memory Decode Circuitry
Static RAMs

APPENDIX B

CONTROLLER SOURCE CODE

```
;
.8080
;
ENTRY   DELAY,TABLE
;
EXTRN   ABORT,WRBUBL,RDBUBL,INBUBL
;
        CSEG
;
;       GAS CAN CONTROL PROGRAM
;
STACK   EQU     0
;
NSC8101 EQU     000H
NSC8102 EQU     020H
NSC8103 EQU     040H
RTC     EQU     060H
BUBBLE  EQU     080H
CONDATA EQU     0A0H
CONSTAT EQU     0C0H
BAUD    EQU     007H
BAUDAD  EQU     001H
POWERCK EQU     001H
ATOD    EQU     0E0H
MONTH   EQU     067H
DAYOFM  EQU     066H
DAYOFW  EQU     065H
HOURS   EQU     064H
MINUTE  EQU     063H
GO      EQU     075H
AMONTH  EQU     06FH
ADAYOFM EQU     06EH
AHOUR   EQU     06CH
AMINUTE EQU     06BH
CR      EQU     0DH
LF      EQU     0AH
BS      EQU     08H
RAM     EQU     0F000H
BUFF    EQU     RAM
TABLE   EQU     0F800H
;
;       JUMP TABLE
;
BOOT:   JMP     SYSTEM
        DS      5
RSTRT1: JMP     INBUBL
        DS      5
RSTRT2: JMP     RDBUBL
        DS      5
RSTRT3: JMP     WRBUBL
        DS      5
RSTRT4: JMP     ABORT
        DS      5
RSTRT5: JMP     DOS
        DS      1
RSTRTC: JMP     DOISS
```

```
        DS      1
RSTRT6: JMP     DO6
        DS      1
RSTRTB: JMP     DO165
        DS      1
RSTRT7: JMP     DO7
        DS      1
RSTRTA: JMP     DO175
        DS      27B
NONMSK: JMP     DONMI
;
;       BEGINNING OF PROGRAM OPERATION
;
SYSTEM:
        DI
        LXI     SP,STACK        ;SET STACK TO 0
        CALL    INITHW          ;INITIALIZE HARDWARE
        LXI     D,MENU          ;PRINT GREETING MENU
        CALL    PRINT
AGAIN:
        LXI     SP,STACK        ;INITIALIZE STACK TO 0
        CALL    CONIN           ;GET INPUT FROM KEYBOARD
        CPI     '0'             ;LOWER LIMIT ERROR ON INPUT
        JM      ERROR
        CPI     '8'             ;UPPER LIMIT ERROR ON INPUT
        JP      ERROR
        SUI     '0'             ;CONVERT FROM ASCII TO HEX
        MOV     C,A             ;DETERMINE POSITION IN JUMP TABLE
        ADD     A
        ADD     C
        MOV     C,A
        XRA     A
        MOV     B,A
        LXI     H,SYSTBL        ;LOADS ADDRESS OF SYSTEM TABLE
        DAD     B               ;FORMULATES JUMP TABLE ADDRESS
        PCHL
;
SYSTBL:                         ;JUMP TABLE
        JMP     SYSTEM          ;GO TO BEGINNING OF PROGRAM
        JMP     DO1             ;REAL TIME CLOCK CONTROL
        JMP     DO2             ;POWER CONTROL
        JMP     DO3             ;INITIALIZE BUBBLE
        JMP     DO4             ;WRITE BUBBLE DATA
        JMP     DO5             ;READ BUBBLE DATA
        JMP     DO6             ;RAM MEMORY
        JMP     DO7             ;ANALOG TO DIGITAL CONVERTOR
;
ERROR:
        LXI     D,MSG1          ;ERROR MESSAGE
        CALL    PRINT
        LXI     D,MENU          ;MAIN MENU
        CALL    PRINT
        JMP     AGAIN           ;GO TO BEGINNING AND GET CONSOLE INPUT
;
;       MAIN PROGRAM
;
```

```
DO1:                                    ;REAL TIME CLOCK
        LXI     D,MSG2                  ;PRINT REAL TIME CLOCK MENU
        CALL    PRINT
        CALL    CONIN                   ;GET CONSOLE INPUT
        CPI     'Ø'                     ;LOWER BOUND INPUT ERROR CHECK
        JM      ERR1
        CPI     '4'                     ;UPPER BOUND INPUT ERROR CHECK
        JP      ERR1
        SUI     'Ø'                     ;ASCII TO HEX CONVERSION
        MOV     C,A                     ;DETERMINE POSITION IN JUMP TABLE
        ADD     A
        ADD     C
        MOV     C,A
        XRA     A
        MOV     B,A
        LXI     H,DO1TBL                ;JUMP TABLE ADDRESS
        DAD     B                       ;CALCULATE POSITION IN JUMP TABLE
        PCHL
;
DO1TBL:                                 ;REAL TIME CLOCK JUMP TABLE
        JMP     DO1Ø                    ;CLEAR INTERRUPT
        JMP     DO11                    ;SET REAL TIME
        JMP     DO12                    ;SET INTERRUPT
        JMP     DO13                    ;SET WAKEUP TIME
;
DO1Ø:                                   ;CLEAR INTERRUPT
        XRA     A
        ADI     ØØH
        OUT     7ØH                     ;INTERRUPT STATUS REGISTER
        JMP     DONE1
;
DO11:                                   ;SET REAL TIME
        LXI     D,MSG11                 ;INPUT MONTH MSG
        CALL    PRINT
        CALL    GETHEX                  ;GET MONTH IN HEX
        XRA     A
        MVI     A,B                     ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     MONTH                   ;LOAD COUNTER WITH MONTH
        LXI     D,MSG12                 ;INPUT DAY OF MONTH MSG
        CALL    PRINT
        CALL    GETHEX                  ;GET DAY OF MONTH MSG
        XRA     A
        MVI     A,B                     ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     DAYOFM                  ;LOAD COUNTER WITH DAY OF MONTH
        LXI     D,MSG13                 ;INPUT DAY OF WEEK
        CALL    PRINT
        CALL    GETHEX                  ;GET DAY OF WEEK IN HEX
        XRA     A
        MVI     A,B                     ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     DAYOFW                  ;LOAD COUNTER WITH DAY OF WEEK
        LXI     D,MSG14                 ;INPUT HOUR OF DAY
        CALL    PRINT
        CALL    GETHEX                  ;GET HOUR OF DAY IN HEX
        XRA     A
        MVI     A,B                     ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     HOURS                   ;LOAD COUNTER WITH HOUR
        LXI     D,MSG15         ;INPUT MINUTE
        CALL    PRINT
        CALL    GETHEX          ;GET MINUTE
        XRA     A
        MVI     A,B             ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     MINUTE          ;LOAD COUNTER WITH MINUTE
        XRA     A
        OUT     GO              ;GO COMMAND
        JMP     DONE1
;
DO12:                           ;SET INTERRUPT
        LXI     D,MSG16         ;INTERRUPT SELECT MENU
        CALL    PRINT
        CALL    CONIN           ;GET INPUT FROM KEYBOARD
        CPI     'Ø'
        JM      ERR1            ;LOWER LIMIT CHECK FOR ERROR ON INPUT
        CPI     '5'
        JP      ERR1            ;UPPER LIMIT CHECK FOR ERROR ON INPUT
        SUI     'Ø'             ;CONVERT FROM ASCII TO HEX
        MOV     C,A             ;CALCULATE TABLE ADDRESS
        ADD     A
        ADD     C
        MOV     C,A
        XRA     A
        MOV     B,A
        LXI     H,CLKTBL        ;LOCATION OF CLOCK TABLE
        DAD     B
        PCHL
;
CLKTBL:                         ;INTERRUPT JUMP TABLE
        JMP     DO111           ;Ø.1 SEC
        JMP     DO112           ;1.Ø SEC
        JMP     DO113           ;1.Ø MIN
        JMP     DO114           ;1.Ø HOUR
        JMP     DO115           ;NO INTERRUPT
;
DO111:                          ;SET INTERRUPT TO Ø.1 SEC
        XRA     A
        MVI     A,Ø2H           ;INTERRUPT OUTPUT TO 1ØHZ
        OUT     Ø71H
        JMP     DONE1
;
DO112:                          ;SET INTERRUPT TO 1.Ø SEC
        XRA     A
        MVI     A,Ø4H           ;INTERRUPT OUTPUT TO 1 HZ
        OUT     Ø71H
        JMP     DONE1
;
DO113:                          ;SET INTERRUPT TO 1 MIN
        XRA     A
        MVI     A,Ø8H           ;INTERRUPT ONCE A MINUTE
        OUT     Ø71H
        JMP     DONE1
;
DO114:                          ;SET INTERRUPT TO 1 HOUR
        XRA     A
        MVI     A,10H           ;INTERRUPT OUTPUT ONCE AN HOUR
        OUT     071H
        JMP     DONE1
;
DO115:                          ;NO INTERRUPT
        XRA     A
        OUT     071H            ;CLEAR INTERRUPTS
        JMP     DONE1
;
DO13:                           ;SET WAKEUP TIME
        LXI     D,MSG11         ;INPUT MONTH MSG
        CALL    PRINT
        CALL    GETHEX          ;GET MONTH FOR RAM
        XRA     A
        MVI     A,B             ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     AMONTH          ;LOAD RAM WITH MONTH
        LXI     D,MSG12         ;INPUT DAY OF MONTH MSG
        CALL    PRINT
        CALL    GETHEX          ;GET DAY OF MONTH FOR RAM
        XRA     A
        MVI     A,B             ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     ADAYOFM         ;LOAD RAM WITH DAY OF MONTH
        XRA     A
        ADI     0FFH
        OUT     06DH
        LXI     D,MSG14         ;INPUT HOUR OF DAY
        CALL    PRINT
        CALL    GETHEX          ;GET HOUR OF DAY FOR RAM
        XRA     A
        MVI     A,B             ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     AHOUR           ;LOAD RAM WITH HOUR
        LXI     D,MSG15         ;INPUT MINUTE
        CALL    PRINT
        CALL    GETHEX          ;GET MINUTE FOR RAM
        XRA     A
        MVI     A,B             ;LOAD ACCUMULATOR WITH HEX VALUE
        OUT     AMINUTE         ;LOAD RAM WITH MINUTE
        JMP     DONE1
;
DONE1:
        LXI     D,MENU          ;PRINT OPENING MENU
        CALL    PRINT
        JMP     AGAIN
;
ERR1:
        LXI     D,MSG1          ;PRINT ERROR MESSAGE
        CALL    PRINT
        JMP     DO1
;
DO2:                            ;UNIT CONTROL MODULE
        LXI     D,MSG6          ;PRINT POWER TO UNIT ON/OFF MESSAGE
        CALL    PRINT
        CALL    CONIN           ;GET KEYBOARD INPUT
        CPI     '0'             ;LOWER BOUND INPUT ERROR CHECK
        JM      ERR2
        CPI     '6'             ;UPPER BOUND INPUT ERROR CHECK
```

```
         JP      ERR2
         SUI     'Ø'             ;ASCII TO HEX CONVERSION
         MOV     C,A             ;CALCULATE POSITION IN JUMP TABLE
         ADD     A
         ADD     C
         MOV     C,A
         XRA     A
         MOV     B,A
         LXI     H,DO2TBL        ;JUMP TABLE ADDRESS
         DAD     B
         PCHL
;
DO2TBL:
         JMP     DO2Ø            ;UNIT #1 OFF (BIT BØ ON NSC81Ø#3)
         JMP     DO21            ;UNIT #1 ON  (BIT BØ ON NSC81Ø#3)
         JMP     DO22            ;UNIT #2 OFF (BIT B1 ON NSC81Ø#3)
         JMP     DO23            ;UNIT #2 ON  (BIT B1 ON NSC81Ø#3)
         JMP     DO24            ;UNIT #3 OFF (BIT B2 ON NSC81Ø#3)
         JMP     DO25            ;UNIT #3 ON  (BIT B2 ON NSC81Ø#3)
;
DO2Ø:
         MVI     A,Ø1H
         OUT     49H             ;TURN UNIT #1 OFF (CLEAR BIT BØ
                                 ;ON NSC81Ø#3)
         JMP     DONE2
;
DO21:
         MVI     A,Ø1H
         OUT     4DH             ;TURN UNIT #1 ON (SET BIT BØ
                                 ;ON NSC81Ø#3)
         JMP     DONE2
;
DO22:
         MVI     A,Ø2H
         OUT     49H             ;TURN UNIT #2 OFF (CLEAR BIT B1
                                 ;ON NSC81Ø#3)
         JMP     DONE2
;
DO23:
         MVI     A,Ø2H
         OUT     4DH             ;TURN UNIT #2 ON (SET BIT B1
                                 ;ON NSC81Ø#3)
         JMP     DONE2
;
DO24:
         MVI     A,Ø4H
         OUT     49H             ;TURN UNIT #3 OFF (CLEAR BIT B2
                                 ;ON NSC81Ø#3)
         JMP     DONE2
;
DO25:
         MVI     A,Ø4H
         OUT     4DH             ;TURN UNIT#3 ON (SET BIT B2
                                 ;ON NSC81Ø#3)
DONE2:
         LXI     D,MENU          ;PRINT OPENING MENU
         CALL    PRINT
         JMP     AGAIN
;
ERR2:
         LXI     D,MSG1          ;PRINT ERROR MESSAGE
         CALL    PRINT
         JMP     DO2
;
DO3:                             ;BUBBLE INITIALIZATION
         LXI     H,TABLE         ;LOAD TABLE ADDRESS
         MVI     M,01H           ;LOAD PARAMETRIC TABLE
         INX     H
         MVI     M,10H
         INX     H
         MVI     M,20H
         INX     H
         MVI     M,00H
         INX     H
         MVI     M,00
         LXI     H,TABLE
         CALL    INBUBL
         ANI     20H
         CPI     20H             ;CHECK FOR SUCCESS INDICATION
         JNZ     DONE3           ;OP-COMPLETE MSG
         LXI     D,MSG5          ;OP-FAIL MESSAGE
         CALL    PRINT
         JMP     DONE31          ;GO TO MAIN MENU
;
DONE3:
         LXI     D,MSG4          ;OP-COMPLETE MSG
         CALL    PRINT
DONE31:
         LXI     D,MENU          ;PRINT MAIN MENU
         CALL    PRINT
         JMP     AGAIN
;
DO4:                             ;WRITE DATA TO BUBBLE
         LXI     H,TABLE         ;LOAD TABLE ADDRESS
         MVI     M,10H
         INX     H
         MVI     M,12H
         INX     H
         MVI     M,20H
         INX     H
         MVI     M,00H
         INX     H
         MVI     M,20H
         LXI     H,TABLE
         LXI     D,BUFF          ;ADDRESS OF RAM BUFFER AREA
         CALL    WRBUBL
         ANI     20H
         CPI     20H             ;CHECK FOR SUCCESSFUL OPS
         JNZ     DONE4           ;OP-COMPLETE MSG
         LXI     D,MSG5          ;OP-FAIL MSG
         CALL    PRINT
         JMP     DONE41
```

```
;
DONE4:
        LXI     D,MSG4          ;OP-COMPLETE MSG
        CALL    PRINT
DONE41:
        LXI     D,MENU          ;PRINT MAIN MENU
        CALL    PRINT
        JMP     AGAIN
;
DO5:                            ;READ DATA FROM BUBBLE
        LXI     H,TABLE         ;LOAD TABLE ADDRESS
        MVI     M,10H
        INX     H
        MVI     M,10H
        INX     H
        MVI     M,20H
        INX     H
        MVI     M,00H
        INX     H
        MVI     M,00H
        LXI     H,TABLE
        LXI     D,BUFF          ;ADDRESS OF RAM BUFFER AREA
        CALL    RDBUBL
        ANI     20H
        CPI     20H             ;CHECK FOR SUCCESSFUL OPS
        JNZ     DONE5           ;OP-COMPLETE MSG
        LXI     D,MSG5          ;OP-FAIL MSG
        CALL    PRINT
        JMP     DONE51
DONE5:
        LXI     D,MSG4          ;OP-COMPLETE MSG
        CALL    PRINT
DONE51:
        LXI     D,MENU          ;PRINT MAIN MENU
        CALL    PRINT
        JMP     AGAIN
;
DOI55:
        LXI     D,MSG7
        CALL    PRINT
        JMP     AGAIN
;
.Z80
;
DO6:                            ;RAM MEMORY TEST PROGRAM
        LD      DE,MSG62        ;RAM BUFFER TEST MENU
        CALL    PRINT
        CALL    CONIN           ;GET CONSOLE INPUT
        CP      '0'             ;LOWER BOUND INPUT ERROR TEST
        JP      M,ERROR6
        CP      '4'             ;UPPER BOUND INPUT ERROR TEST
        JP      P,ERROR6
        SUB     '0'             ;ASCII TO HEX CONVERSION
        LD      C,A             ;CALCULATE POSITION IN JUMP TABLE
        ADD     A,A
        ADD     A,C
```

```
          LD      C,A
          XOR     A
          LD      B,A
          LD      HL,BUFFTBL          ;LOAD JUMP TABLE ADDRESS
          ADD     HL,BC               ;CALCULATE JUMP ADDRESS
          JP(HL)
;
BUFFTBL:                              ;INDEX ADDRESS LOCATION
          JP      D061                ;INITIALIZE BUFFER WITH Ø'S
          JP      D062                ;LOAD BUFFER MEMORY WITH CHARACTER
          JP      DC63                ;RETURN TO SYSTEM
          JP      D064
;
ERROR6:
          LD      DE,MSG1             ;BAD INPUT MSG
          CALL    PRINT
          JP      DC6
;
D061:                                 ;INITIALIZE BUFFER TO Ø
          LD      A,Ø                 ;CLEAR A
          LD      HL,BUFF             ;INITIALIZE BUFFER
LOOP61:   LD      B,Ø
SPOT1:    LD      (HL),3ØH
          INC     HL
          DJNZ    SPOT1
          INC     A
          CP      4
          JP      Z,DONE61
          JP      LOOP61
;
DC62:                                 ;LOAD BUFFER MEMORY
          LD      DE,MSG6Ø            ;MSG TO HAVE USER
                                      ;DEPRESS KEY
                                      ;KEY ON TERMINAL TO LOAD
                                      ;CHARACTER INTO MEMORY
          CALL    PRINT               ;WRITE MESSAGE ON TERMINAL
          CALL    CONIN
          CP      CR
          JP      Z,DONE62            ;IF CR END TEST
          LD      C,A
          LD      A,Ø                 ;CLEAR A
          LD      HL,BUFF
LOOPS:    LD      B,Ø
SPOT:     LD      (HL),C              ;LOAD MEMORY WITH TERMINAL VALUE
          INC     HL
          DJNZ    SPOT
          INC     A
          CP      4
          JP      Z,DONE61            ;LOAD 1024 MEMORY LOCATIONS
          JP      LOOPS
DONE61:
          LD      DE,MSG4
          CALL    PRINT
DONE62:
          JP      D06
;
```

```
D063:                                 ;RAM MEMORY VIEW MODULE
        LD      A,Ø                   ;INITIALIZE A
        LD      HL,BUFF
LOOP3:  LD      B,Ø
SPOT3:  LD      C,(HL)
        PUSH    AF
        CALL    CONOUT
        POP     AF
        INC     HL
        DJNZ    SPOT3
        INC     A
        CP      4
        JP      Z,DONE61
        JP      LOOP3
;
D064:                                 ;RETURN TO SYSTEM
        LD      DE,MENU
        CALL    PRINT
        JP      AGAIN
;
.8Ø8Ø
;
DOI65:
        LXI     D,MSG7
        CALL    PRINT
        JMP     AGAIN
;
D07:
        LXI     D,MSG71
        CALL    PRINT
        CALL    CONIN                 ;GET INPUT FROM KEYBOARD
        CPI     'Ø'
        JM      ADERR                 ;LOWER LIMIT CHECK ON INPUT
        CPI     '5'
        JP      ADERR                 ;UPPER LIMIT CHECK ON INPUT
        SUI     'Ø'                   ;CONVERT FROM ASCII TO HEX
        MOV     C,A                   ;CALCULATE TABLE ADDRESS
        ADD     A
        ADD     C
        MOV     C,A
        XRA     A
        MOV     B,A
        LXI     HL,ADTBL              ;LOCATION OF A-TO-D TABLE
        DAD     B
        PCHL
;
ADTBL:                                ;A-TO-D JUMP TABLE
        JMP     D0711                 ;CHANNEL Ø
        JMP     D0712                 ;CHANNEL 1
        JMP     D0713                 ;CHANNEL 2
        JMP     D0714                 ;CHANNEL 3
        JMP     D0715                 ;CHANNEL 4
;
D0711:  IN      ØEØH                  ;CHANNEL Ø INPUT
        JMP     ADEND
D0712:  IN      ØE1H                  ;CHANNEL 1 INPUT
```

```
        JMP     ADEND
DO713:  IN      0E2H            ;CHANNEL 2 INPUT
        JMP     ADEND
DO714:  IN      0E3H            ;CHANNEL 3 INPUT
        JMP     ADEND
DO715:  IN      0E4H            ;CHANNEL 4 INPUT
ADEND:  LXI     D,MENU
        CALL    PRINT
        JMP     AGAIN
;
ADERR:  LXI     D,MSG72         ;ERROR ON INPUT
        CALL    PRINT
        JMP     DO7
;
DOI75:
        LXI     D,MSG7
        CALL    PRINT
        JMP     AGAIN
;
DONMI:
        LXI     D,MSG7
        CALL    PRINT
        JMP     AGAIN
;
;       START OF SUBROUTINES
;
PRINT:
        XCHG                    ;SWAP BC REG WITH HL REG
PRT1:
        MOV     A,M             ;MOVE MEMORY VALUE TO A REG
        CPI     '$'             ;CHECK FOR END OF MSG DELIMITER
        RZ                      ;RETURN IF END OF MSG FOUND
        MOV     C,A
PRT2:
        IN      CONSTAT         ;CHECK FOR CONSOLE STATUS
        ANI     01              ;MASK OUT ALL BITS EXCEPT BIT 0
        CPI     01              ;CHECK TO SEE IF TRANSMIT BUFFER IS EMPTY
        JNZ     PRT2
        MOV     A,C             ;LOAD CHARACTER TO BE TRANSMITTED
        OUT     CONDATA         ;SEND DATA TO TERMINAL
        INX     H
        JMP     PRT1            ;CHECK NEXT CHARACTER
;
CONIN:
        IN      CONSTAT         ;CHECK UART STATUS
        ANI     02              ;CHECK TO SEE IF INPUT HAS BEEN MADE
        JZ      CONIN
        IN      CONDATA         ;RECEIVE DATA FROM UART
        ANI     7FH             ;MASK OUT HIGH BIT
        RET
;
CONOUT:
        IN      CONSTAT
        ANI     01
        CPI     01
        JNZ     CONOUT
        MOV     A,C
        OUT     CONDATA
        RET
;
CONST:
        IN      CONSTAT
        ANI     Ø2
        RZ
        MVI     A,ØFFH
        RET
;
INITHW:
        MVI     A,Ø1H
        OUT     Ø7H             ;SET NSC81Ø#1 MODE DEF REG FOR
                                ;PORT A TO STROBED MODE INPUT
                                ;FOR POWER CONTROL OPS
        MVI     A,ØØH
        OUT     Ø4H             ;DATA DIRECTION REGISTERS ON PORT
                                ;A SET AS INPUT
        MVI     A,Ø3H
        OUT     Ø6H             ;DATA DIRECTION REGISTER ON PORT C
                                ;SET AS INPUT FOR STROBED POWER
                                ;CONTROL OPERATION
        MVI     A,Ø4H
        OUT     ØEH             ;ENABLE THE ACTIVE-LOW INTERRUPT
                                ;FROM THE IO TO THE CPU
        MVI     A,ØFFH
        OUT     Ø5H             ;PORT B CONFIGURED AS AN OUTPUT
                                ;FOR UART CONTROL
        MVI     A,ØØH
        OUT     19H             ;TIMER1 IS STOPPED AND RESET
        MVI     A,Ø5H
        OUT     19H             ;TIMER1 SET TO SQUARE WAVE
        MVI     A,BAUDAD
        OUT     12H             ;TIMER1 LB A-TO-D BAUD RATE
        MVI     A,ØØH
        OUT     13H             ;TIMER1 HB A-TO-D BAUD RATE
        OUT     17H             ;START TIMER1
        MVI     A,ØØH
        OUT     18H             ;TIMERØ IS STOPPED AND RESET
        MVI     A,Ø5H
        OUT     18H             ;TIMERØ SET TO SQUARE WAVE
        MVI     A,BAUD
        OUT     1ØH             ;TIMERØ LB UART BAUD RATE SET
        MVI     A,ØØH
        OUT     11H             ;TIMERØ HB UART BAUD RATE SET
        OUT     15H             ;START TIMER
        MVI     A,Ø1H
        OUT     27H             ;SET NSC81Ø#2 MODE DEFINE REGISTER
                                ;FOR STROBED SSDR INPUT TO PORT A
        MVI     A,ØØH
        OUT     24H             ;DATA DIRECTION REGISTER ON PORT
                                ;A SET AS INPUT
        MVI     A,Ø3H
        OUT     26H             ;DIR PORT C SET UP FOR STROBED INPUT
                                ;FOR SSDR HANDSHAKING
```

```
            MVI     A,04H
            OUT     2EH                 ;ENABLE THE ACTIVE-LOW INTERRUPT
                                        ;FROM THE I/O TO THE CPU
            MVI     A,00H
            OUT     25H                 ;PORT B IS CONFIGURED AS POWER
                                        ;GROUP INPUT
            MVI     A,00H
            OUT     38H                 ;TIMER0 IS STOPPED AND RESET
            MVI     A,05H
            OUT     38H                 ;TIMER0 SET TO SQUARE WAVE
            MVI     A,POWERCK
            OUT     30H                 ;TIMER0 LB POWER CLOCK SET
            MVI     A,00H
            OUT     31H                 ;TIMER0 HB POWER CLOCK SET
            OUT     35H                 ;START TIMER
            MVI     A,07H
            OUT     47H                 ;SET NSC810#3 MODE DEFINE REGISTER
                                        ;FOR STROBED SSDR OUTPUT (TRI-STATE)
                                        ;ON PORT A
            MVI     A,0FFH
            OUT     44H                 ;DATA DIRECTION REGISTER ON PORT A
                                        ;SET AS OUTPUT
            MVI     A,03H
            OUT     46H                 ;DDR PORT C SET UP FOR STROBED
                                        ;OUTPUT (TRI-STATE) FOR SSDR
                                        ;HANDSHAKING
            MVI     A,04H
            OUT     4EH                 ;ENABLES THE ACTIVE-LOW INTERRUPT
                                        ;FROM THE I/O TO THE CPU
            MVI     A,0FFH
            OUT     45H                 ;PORT B CONFIGURED AS POWER
                                        ;GROUP OUTPUT
            MVI     A,00H
            OUT     58H                 ;TIMER0 STOPPED AND RESET
            MVI     A,05H
            OUT     58H                 ;TIMER0 SET TO SQUARE WAVE
            MVI     A,POWERCK
            OUT     50H                 ;TIMER0 LB POWER GROUP TIME SET
            MVI     A,00H
            OUT     51H                 ;TIMER0 HB POWER GROUP TIME SET
            OUT     55H                 ;START TIMER0
            RET
;
GETHEX:                                 ;SUBROUTINE TO GET A 2 DIGIT NUMBER
                                        ;FROM THE KEYBOARD. CONVERT THEM TO
                                        ;HEX AND THEN TO A TWO DIGIT BCD
                                        ;NUMBER
HIGH:       LXI     D,MSG111            ;MSG TO INPUT HIGH DIGIT
            CALL    PRINT
            CALL    CONIN               ;GET HIGH DIGIT FROM KEYBOARD
            CPI     '0'
            JM      ERRHI1              ;LOWER LIMIT CHECK ON INPUT
            JMP     CONT1               ;CORRECT INPUT - CONTINUE
ERRHI1:     LXI     D,MSG113            ;ERROR MSG ON INPUT
            CALL    PRINT
            JMP     HIGH
```

```
CONT1:  CPI     3AH
        JP      ERRHI2          ;HIGH LIMIT CHECK ON INPUT
        JMP     CONT2           ;CORRECT INPUT - CONTINUE
ERRHI2: LXI     D,MSG113        ;ERROR MSG ON INPUT
        CALL    PRINT
        JMP     HIGH
CONT2:  SUI     'Ø'             ;CONVERT FROM ASCII TO HEX
        RAL
        RAL                     ;THIS MOVES THE HEX VALUE TO BIT
        RAL                     ;POSITIONS 4 TO 7 IN THE ACCUMULATOR
        RAL
        ANI     ØFØH            ;THIS ZEROS THE LOWER 4 BITS IN
                                ;THE ACCUMULATOR
        MOV     C,A             ;DIGIT IS PLACED IN C REG
LOW:    LXI     D,MSG112        ;MESSAGE TO INPUT LOW DIGIT
        CALL    PRINT
        CALL    CONIN           ;GET LOW DIGIT
        CPI     'Ø'
        JM      ERRLOW1         ;LOWER LIMIT CHECK ON INPUT
        JMP     CONT3           ;CORRECT INPUT - CONTINUE
ERRLOW1:
        LXI     D,MSG113        ;ERROR ON INPUT
        CALL    PRINT
        JMP     LOW
CONT3:  CPI     3AH
        JP      ERRLOW2         ;HIGH LIMIT CHECK ON INPUT
        JMP     CONT4
ERRLOW2:
        LXI     D,MSG113        ;ERROR ON INPUT
        CALL    PRINT
        JMP     LOW
CONT4:  SUI     'Ø'             ;CONVERT FROM ASCII TO HEX
        ANI     ØFH             ;THIS ZEROS THE HIGH 4 BITS OF
                                ;THE ACCUMULATOR
        ORA     C               ;THIS JOINS TOGETHER THE BCD PAIR
        MOV     B,A             ;THE BCD PAIR IS MOVED TO REG B
        RET
;
DELAY:
        MOV     B,A     ;DELAY A TIMES 1ØMSEC.
LOOP1:
        LXI     D,1Ø41
LOOP2:
        DCX     D
        MOV     A,D
        ORA     E
        JNZ     LOOP2
        DCR     B
        JNZ     LOOP1
        RET
;
;       MESSAGES
;
MENU:   DB      CR,LF,'GAS CAN CONTROL PROGRAM',CR,LF
```

```
        DB      CR,LF,'0= RESET SYSTEM '

        DB      CR,LF,'1= REAL TIME CLOCK '

        DB      CR,LF,'2= POWER CONTROL '

        DB      CR,LF,'3= INITIALIZE BUBBLE '

        DB      CR,LF,'4= WRITE BUBBLE DATA '

        DB      CR,LF,'5= READ BUBBLE DATA '

        DB      CR,LF,'6= MEMORY BUFFER

        DB      CR,LF,'7- A TO D CONVERTER '

        DB      CR,LF,'$'
;
MSG1:   DB      CR,LF,'BAD ENTRY, TRY AGAIN!',CR,LF,'$'
```

```
;
MSG2:   DB      CR,LF,'SET REAL TIME CLOCK ',CR,LF

        DB      CR,LF,'0= CLEAR INTERRUPT '

        DB      CR,LF,'1= SET REAL TIME '

        DB      CR,LF,'2= SET INTERRUPT '

        DB      CR,LF,'3= SET WAKEUP TIME '

        DB      CR,LF,'$'
;
MSG3:   DB      CR,LF,'SPARE ',CR,LF,'$'
;
MSG4:   DB      CR,LF,'OP-COMPLETE ',CR,LF '$'
;
MSG5:   DB      CR,LF,'OP-FAILED ',CR,LF,'$'
;
MSG6:   DB      CR,LF,'SELECT UNIT TO TURN ON/OFF',CR,LF
```

```
        DB      CR,LF,'0= UNIT #1 OFF'

        DB      CR,LF,'1= UNIT #1 ON'

        DB      CR,LF,'2= UNIT #2 OFF'

        DB      CR,LF,'3= UNIT #2 ON'

        DB      CR,LF,'4= UNIT #3 OFF'

        DB      CR,LF,'5= UNIT #3 ON'

        DB      CR,LF,'$'
;
MSG7:   DB      CR,LF,'SPURIOUS INTERRUPT ',CR,LF,'$'


;
MSG11:  DB      CR,LF,'INPUT MONTH = ',CR,LF,'$'


;
MSG12:  DB      CR,LF,'INPUT DAY OF MONTH = ',CR,LF,'$'


;
MSG13:  DB      CR,LF,'INPUT DAY OF WEEK = ',CR,LF,'$'


;
MSG14:  DB      CR,LF,'INPUT HOUR OF DAY = ',CR,LF,'$'
```

```
;
MSG15:  DB      CR,LF,'INPUT MINUTE OF HOUR = ',CR,LF,'$
;
MSG16:  DB      CR,LF,'SET INTERRUPT INTERVAL',CR,LF,CR,LF

        DB      '0 = 0.1 SEC',CR,LF

        DB      '1 = 1.0 SEC',CR,LF

        DB      '2 = 1.0 MIN',CR,LF

        DB      '3 = 1.0 HOUR',CR,LF

        DB      '4 = NO INTERRUPT',CR,LF

        DB      '$'
;
MSG60:  DB      CR,LF,'DEPRESS ANY LETTER/NUMBER',CR,LF

        DB      CR,LF,'OR CR TO QUIT',CR,LF
        DB      CR,LF,'$'
;
MSG62:  DB      CR,LF,'RAM MEMORY TEST OPTION:' CR LF

        DB      CR,LF,'         ',CR,LF

        DB      CR,LF,'0 = INITIALIZE BUFFER MEMORY',CR,LF

        DB      CR,LF,'1 = LOAD BUFFER MEMORY ,CR,LF

        DB      CR,LF,'2 = DISPLAY BUFFER MEMORY' CR,LF

        DB      CR,LF,'3 = RETURN TO SYSTEM',CR,LF

        DB      CR,LF,'$'
;
MSG71:  DB      CR,LF,'SELECT CHANNEL TO READ ON THE',CR,LF

        DB      'ANALOG-TO-DIGITAL CONVERTER:',CR,LF
        DB      CR,LF,'0 = CHANNEL 0',CR,LF

        DB      '1 = CHANNEL 1',CR,LF

        DB      '2 = CHANNEL 2',CR,LF

        DB      '3 = CHANNEL 3',CR,LF

        DB      '4 = CHANNEL 4',CR,LF

        DB      '$'
;
MSG72:  DB      CR,LF,'ERROR ON INPUT - PLEASE SELECT A' CR,LF

        DB      'DIGIT FROM 0 TO 4 ONLY',CR,LF

        DB      '$'
;
MSG111: DB      CR,LF,'ENTER HIGH DIGIT OR 0 IF NOT ,CR,LF

        DB      'APPLICABLE',CR,LF
        DB      '$'
;
MSG112: DB      CR,LF,'ENTER LOW DIGIT',CR LF

        DB      '$'
;
MSG113: DB      CR,LF,'PLEASE - ONLY ENTER DIGITS FROM 0 TO 9 ',CR,LF

        DB      '$'
:
        DS      1
;
        END
```

```
;
.8Ø8Ø
;
PRTAØØ  EQU     Ø8ØH            ;POLLED MODE
PRTAØ1  EQU     Ø81H            ;I/O PORTS
BYTCNT  EQU     1Ø24            ;RAM BUFFER BLOCK
;
ENTRY   ABORT,WRBUBL,RDBUBL,INBUBL
EXTRN   DELAY,TABLE
;
;       INITIALIZE THE PARAMETRIC REGISTERS
;
;       THIS WILL DESTROY THE A AND F/FS
;
INTPAR: PUSH    B
        PUSH    D
        MVI     A,ØBF
        OUT     PRTAØ1          ;ADDRESS LOADED INTO 722Ø RAC
        MVI     D,Ø5H           ;LOOP COUNTER INTIALIZED TO
                                ;READ TABLE VALUE
LOAD:   LDAX    B
        OUT     PRTAØØ          ;WRITE PARAMETRIC REG
        INX     B               ;INCREMENT B-C REGS ADDRESS IN RAM
        DCR     D               ;DECREMENT LOOP COUNTER
        JNZ     LOAD            ;JUMP TO LOAD IF NOT Ø
        POP     D
        POP     B
        RET
;
;       RESET 722Ø FIFO DATA BUFFER
;
;       THIS WILL DESTROY A AND F/FS
;
FIFORS: PUSH    D
        PUSH    B
        MVI     B,4ØH           ;LOAD OP-COMPLETE
        LXI     D,ØFFFFH        ;TIME OUT IS INTIALIZED
        MVI     A,1DH
        OUT     PRTAØ1          ;WRITE FIFO RESET COMMAND
BUSYFR: IN      PRTAØ1          ;READ STATUS REG
        RLC                     ;TEST BUSY BIT=1
        JC      POLLFR          ;IF BUSY=1,POLL STATUS REG
        DCX     D               ;DECREMENT TIME OUT LOOP
        XRA     A
        ORA     D               ;TEST D-REG=ØØH
        ORA     E               ;TEST E-REG=ØØH
        JNZ     BUSYFR          ;IF NOT Ø CONTINUE POLLING
        JMP     RETFR           ;TIME OUT ERROR
;
POLLFR: IN      PRTAØ1          ;READ STATUS
        XRA     B               ;TEST STATUS FOR OP-COMPLETE
        JZ      RETFR           ;JUMP TO RETFR IF OP-COMPLETE
        DCX     D               ;DECREMENT TIME  OUT LOOP
        XRA     A
        ORA     D               ;TEST D-REG
        ORA     E               ;TEST E-REG
```

```
         JNZ     POLLRR            ;IF NOT 0 CONTINUE POLLING
RETRR:   POP     B
         POP     D
         IN      PRTA01            ;READ STATUS
         RET
;
;        WRITE TO BUBBLE MEMORY
;
;        DESTROYS A, H, L, AND F/FS REGS
;
WRITE:   PUSH    D                 ;SAVE RAM BUFFER ADDRESS
         PUSH    B                 ;SAVE TABLE ADDRESS
         LXI     B,0FFFFH          ;INIT TIME OUT LOOP COUNTER
         MVI     A,13H
         OUT     PRTA01            ;WRITE, WRITE BUBBLE MEM DATA CMD
BUSYWR:  DCX     B                 ;DECREMENT TIME OUT LOOP COUNTER
         XRA     A
         ORA     B                 ;TEST B REG=00H
         ORA     C                 ;TEST C REG=00H
         JZ      FINSHW            ;IF 0, TIME OUT ERROR  JMP FINSHW
         IN      PRTA01            ;READ STATUS REG
         RLC                       ;TEST BUST BIT=1
         JNC     BUSYWR            ;IF 0. CONT POLLING BUSY BIT
POLLWR:  IN      PRTA01            ;READ STATUS REG
         RRC                       ;TEST FIFO READY BIT=1
         JC      WFIFO             ;IF FIFO READY=1 JMP WFIFO
         IN      PRTA01            ;READ STATUS REG
         RLC                       ;TEST BUST BIT=1
         JNC     FINSHW            ;IF 0. ERROR, JMP FINSHW
         DCX     B                 ;DEC TIME OUT LOOP COUNTER
         XRA     A
         ORA     B                 ;TEST B REG=00H
         ORA     C                 ;TEST C REG=00H
         JZ      FINSHW            ;IF 0. TIME OUT ERROR  JMP FINSHW
         JMP     POLLWR            ;CONTINUE POLLING FIFO READY BIT
;
WFIFO:   LDAX    D                 ;LOAD RAM ADDRESS
         OUT     PRTA00            ;WRITE A REG TO 7220 FIFO DATA BUFFER
         INX     D                 ;INC TO NEXT ADDRESS IN RAM
         DCX     H                 ;DEC BYTE COUNTER
         XRA     A
         ORA     H                 ;TEST H REG= 00H
         ORA     L                 ;TEST L REG= 00H
         JNZ     POLLWR            ;IF BYTE CTR NOT 0  JMP POLLWR
FINSHW:  POP     B
         POP     D
         RET                       ;RETURN TO CALL
;
;        READ BUBBLE MEMORY
;
;        WILL DESTROY A,H,L, AND F/FS REGS
;
READ:    PUSH    D                 ;SAVE RAM BUFFER ADDRESS
         PUSH    B                 ;SAVE TABLE ADDRESS
         LXI     B,0FFFFH          ;INIT TIME OUT LOOP COUNTER
         MVI     A,12H
```

```
        OUT     PRTA01          ;WRITE, READ BUBBLE MEM DATA CMD
BUSYRD: DCX     B               ;DEC TIME OUT LOOP COUNTER
        XRA     A
        ORA     B               ;TEST B REG= 00H
        ORA     C               ;TEST C REG= 00H
        JZ      FINSHR          ;IF 0, TIME OUT ERROR, JMP FINSHR
        IN      PRTA01          ;READ STATUS REG
        RLC                     ;TEST BUSY BIT= 1
        JNC     BUSYRD          ;IF 0, CONT POLLING BUSY BIT
POLLRD: IN      PRTA01          ;READ STATUS REG
        RRC                     ;TEST FIFO READY BIT= 1
        JC      RFIFO           ;IF FIFO READY= 1, JMP RFIFO
        IN      PRTA01          ;READ STATUS REG
        RLC                     ;TEST BUSY BIT=1
        JNC     FINSHR          ;IF 0, ERROR, JMP FINSHR
        DCX     B               ;DEC TIME OUT LOOP COUNTER
        XRA     A
        ORA     B               ;TEST B REG= 00H
        ORA     C               ;TEST C REG= 00H
        JZ      FINSHR          ;IF 0, TIME OUT ERROR. JMP FINSHR
        JMP     POLLRD          ;CONT POLLING FIFO READY BIT.
;
RFIFO:  IN      PRTA00          ;LOAD A REG W/ 1 BYTE FM FIFO DATA BUFFER
        STAX    D               ;STORE A REG IN REG D-E ADDRESS
        INX     D               ;INC D-E REG TO NEXT ADDR IN RAM
        DCX     H               ;DEC BYTE COUNTER
        XRA     A
        ORA     H               ;TEST H REG= 00H
        ORA     L               ;TEST L REG= 00H
        JNZ     POLLRD          ;IF BYTE COUNTER NOT 0, JMP POLLRD
FINSHR: POP     B               ;RESTORE B-C REGS
        POP     D               ;RESTORE D-E REGS
        RET
;
;       ABORT
;
;       WILL DESTROYS A, AND F/FS REGS
;
ABORT:  PUSH    D               ;PUSH UNKNOWN VALUE OF D TO STACKL
        PUSH    B               ;40H PLACED ON STACK
        LXI     D,0FFFFH        ;INIT TIME OUT LOOP COUNTER
        MVI     B,40H           ;LOAD OPERATION COMPLETE
        MVI     A,19H           ;LOAD ABORT COMMAND
        OUT     PRTA01          ;WRITE ABORT COMMAND
BUSYA:  IN      PRTA01          ;READ STATUS REG
        RLC                     ;CHECK IF BUSY
        JC      POLLA           ;IF BUSY, POLL STATUS REGISTER
        DCX     D               ;DEC TIME OUT COUNTER
        XRA     A
        ORA     D               ;TEST D REG= 00H
        ORA     E               ;TEST E REG= 00H
        JNZ     BUSYA           ;CHECK FOR BUSY IF TIME LEFT
        JMP     RETA            ;TIME OUT ERROR, RETURN
;
POLLA:  IN      PRTA01          ;READ STATUS REG
        XRA     B               ;TEST FOR OP-COMPLETE
        JZ      RETA            ;RETURN IF OP COMPLETE
        DCX     D               ;DEC TIME OUT LOOP COUNTER
        XRA     A
        ORA     D               ;TEST D REG FOR 0
        ORA     E               ;TEST E REG FOR 0
        JNZ     POLLA           ;IF NOT 0 CONTINUE POLLING
RETA:   POP     E
        POP     D
        IN      PRTA01          ;READ STATUS
        RET
;
;       WRITE BUBBLE MEMORY DATA
;
;       WILL DESTROY A, AND F/FS REGS
;
WRBUBL: PUSH    E               ;SAVE END TABLE ADDRESS
        PUSH    B               ;SAVE BEGINNING TABLE ADDRESS
        MVI     B,40H           ;LOAD B REG OP-COMPLETE
        CALL    FIFORS          ;RESET FIFO
        XRA     B               ;TEST FOR OP-COMPLETE
        JNZ     RETWR           ;IF ERROR JMP RETWR
        POP     B
        CALL    INTPAR          ;LOAD PARAMETRIC REGS
        LXI     H,BYTCNT
        CALL    WRITE
        PUSH    B
        LXI     H,0FFFFH        ;INITIALIZE TIME OUT LOOP
LOOPWR: IN      PRTA01          ;READ STATUS
        RLC                     ;TEST   FOR BUSY=1
        JNC     RETWR           ;IF ZERO JMP RETWR
        DCX     H               ;DECREMENT TIME OUT LOOP
        XRA     A
        ORA     H               ;TEST H-REG FOR 0
        ORA     L               ;TEST L-REG FOR 0
        JNZ     LOOPWR          ;CONTINUE POLLING
RETWR:  POP     B
        POP     E
        IN      PRTA01          ;READ STATUS
        RET
;
;       READ BUBBLE MEMORY DATA
;
;       WILL DESTROY A, AND F/FS REGS
;
RDBUBL: PUSH    E               ;END TABLE ADDRESS
        PUSH    B               ;BEGINNING TABLE ADDRESS
        MVI     B,40H           ;LOAD OP-COMPLETE
        CALL    FIFORS          ;RESET FIFO
        XRA     B               ;TEST FOR OP-COMPLETE
        JNZ     RETRD           ;IF NOT ZERO JMP RETRD
        POP     B
        CALL    INTPAR          ;LOAD PARAMETRIC REGS
        LXI     H,BYTCNT
        CALL    READ            ;READ BUBBLE DATA
        PUSH    B
        LXI     H,0FFFFH        ;INITIALIZE TIME OUT LOOP
```

```
LOOPRD: IN      PRTA21          ;READ STATUS
        RLC                     ;TEST FOR BUSY=1
        JNC     RETRD           ;IF ZERO,NOT BUSY,JMP RETRD
        DCX     H               ;DECREMENT TIME OUT LOOP
        XRA     A
        ORA     H               ;TEST H REG=Ø
        ORA     L               ;TEST L REG=Ø
        JNZ     LOOPRD          ;CONTINUE POLLING
RETRD:  POP     D
        POP     H
        IN      PRTAØ1          ;READ STATUS
        RET
;
;       INITIALIZE THE BUBBLE
;
;       WILL DESTROY A, AND F/FS REGS
;
INBUBL: PUSH    D               ;PUSH UNKNOWN VALUE
        PUSH    B               ;ADDRESS OF TABLE PUSHED TO STACK
        MVI     B,40H           ;LOAD OP-COMPLETE
        CALL    ABORT           ;CALL ABORT COMMAND
        XRA     B               ;TEST FOR OP-COMPLETE
        JNZ     RETIN           ;IF ZERO OP-COMPLETE
        POP     B               ;PLACE ADDRESS OF PARAMETRIC REG IN B
        CALL    INTPAR          ;LOAD PARAMETRIC REGS
        PUSH    B
        MVI     B,40H           ;LOAD OP-COMPLETE
        LXI     D,ØFFFFH        ;INITIALIZE TIME OUT LOOP
        MVI     A,11H
        OUT     PRTAØ1          ;WRITE INITIALIZE COMMAND
BUSYIN: IN      PRTAØ1          ;READ STATUS
        RLC                     ;DECREMENT TIME OUT LOOP
        JC      POLLIN          ;IF BUSY=1 POLL FOR 40H
        DCX     D               ;DECREMENT TIME OUT LOOP
        XRA     A
        ORA     D               ;TEST D REG FOR Ø
        ORA     E               ;TEST E REG FOR Ø
        JNZ     BUSYIN          ;IF NOT Ø CONTINUE POLLING
        JMP     RETIN           ;TIME OUT ERROR, RETURN
;
POLLIN: IN      PRTA21          ;READ STATUS
        XRA     B               ;TEST FOR OP-COMPLETE
        JZ      RETIN           ;IF OP-COMPLETE JMP RETIN
        DCX     D               ;DECREMENT TIME OUT LOOP
        XRA     A
        ORA     D               ;TEST D REG FOR Ø
        ORA     E               ;TEST E REG FOR Ø
        JNZ     POLLIN          ;IF NOT Ø CONTINUE POLLING
;
RETIN:  POP     B               ;TABLE ADDRESS GOES TO B
        POP     D               ;RESTORE D E REGS
        IN      PRTA21          ;READ STATUS REG
        RET
;
        DS      1
        END
```

LIST OF REFERENCES

1. Snider, M., NSC888_Self-Contained_NSC800_Evaluation System, M. S. Thesis, Naval Postgraduate School, Monterey, California, December 1984.

2. NASA, Get_Away_Special_(GAS)_Small_Self-Contained Payloads_Experimenter_Handbook, Goddard Space Flight Center Special Payloads Division, July 1984.

3. National Semiconductor Corporation, NSC800_High-Performance_Low-Power_Microprocessor_Data_Sheet, Santa Clara, California, pp. 1 - 27, July 1983.

4. National Semiconductor Corporation, NSC810A_RAM-I/O-Timer, Santa Clara, California, pp. 1 - 17, February 1984.

5. Intel Corporation, BPK_72_Bubble_Memory_Prototype_Kit User's_Manual, Santa Clara, California, pp. 1-1 - 8-12, June 1983.

6. National Semiconductor Corporation, ADC0816_Single Chip_Acquisition_System, Santa Clara, California, pp. 2-29 - 2-38, 1981.

7. Intersil Corporation, IM6402_Universal_Asynchronous Receiver_Transmitter_(UART), Cupertino, California, pp. 2-3 - 2-10, 1983.

8. Intersil Corporation, IM6402_Universal_Asynchronous Receiver_Transmitter_(UART), Cupertino, California, p. 2-8, 1983.

9. National Semiconductor Corporation, MM58167A Microprocessor_Real_Time_Clock, Santa Clara, California, pp. 3-11 - 3-18.

10. National Semiconductor Corporation, MM58167A Microprocessor_Real_Time_Clock, Santa Clara, California, p. 3-12.

11. Hitachi Corporation, HM6116_2048-Word_X_8-Bit_High Speed_Static_CMOS_RAM, pp. 66 - 69.

12. Intel Corporation, 2732A_32K_(4K_X_8)_UV_Erasable PROM, Santa Clara, California, pp. 4-12 - 4-18, 1982.

13. Intel Corporation, BPK_72_Bubble_Memory_Prototype_Kit

User's Manual, Santa Clara, California, p. 5-13, June 1983.

14. Intel Corporation, BPK 72 Bubble Memory Prototype Kit User's Manual, Santa Clara, California, pp. 5-1 - 5-36, June 1983.

15. Dercole, T. and Frey, T., Solid-State Digital Recorder, ECE-3800 Project, Naval Postgraduate School, Monterey, California, December 1984.

16. Intel Corporation, BPK 72 Bubble Memory Prototype Kit User's Manual, Santa Clara, California, p. 5-32, June 1983.

17. Intel Corporation, BPK 72 Bubble Memory Prototype Kit User's Manual, Santa Clara, California, p. 5-34, June 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
4.	Professor R. Panholzer, Code 62Pz Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
5.	Professor C. H. Lee, Code 62Le Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
6.	Professor S. Garrett, Code 61Gx Department of Physics Naval Postgraduate School Monterey, California 93943	1
7.	Lieutenant Jay W. Wallin, USN 5616 Madeira Pl. N.E. Albuquerque, New Mexico 87112	1

END

FILMED

3-86

DTIC